

Actividad 4 : Pruebas de particionamiento de bases de datos NoSQL

Laura Daniela Romero Montañez

Facultad de ingeniería, Universidad Iberoamericana

Ingeniería en Software

Licenciado: WILLIAM RUIZ

Abril del 2023

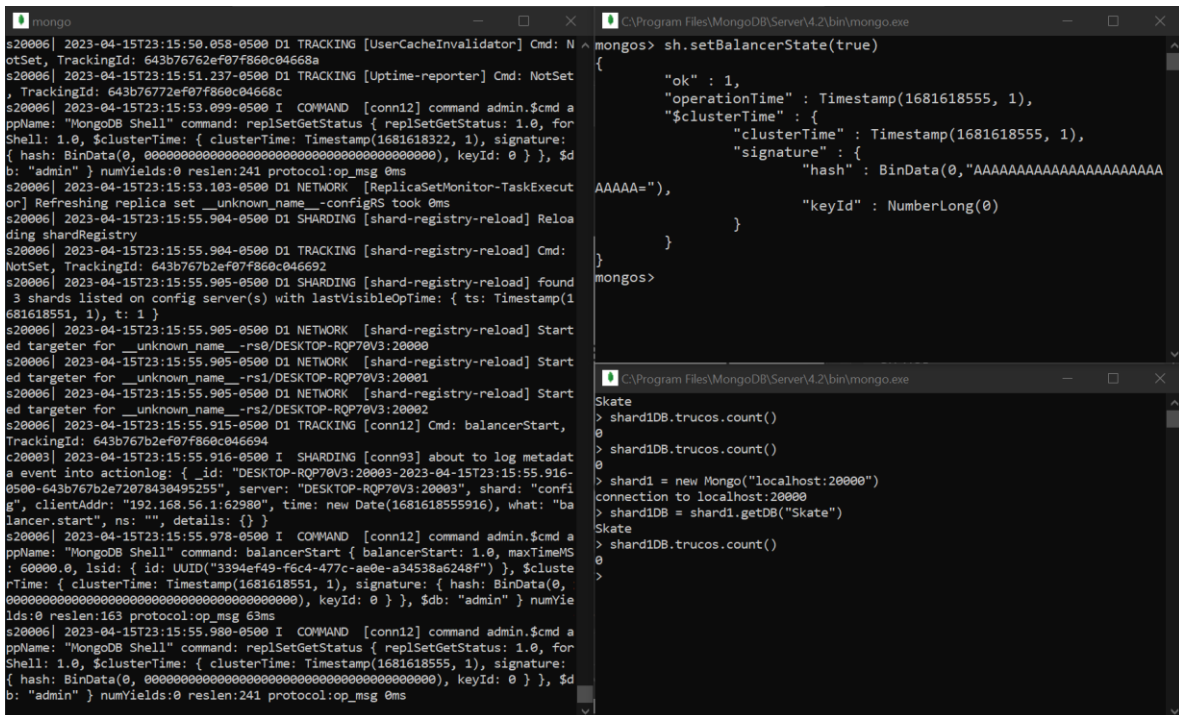
Requerimientos no funcionales

- Distribución uniforme de la carga: las particiones deben permitir una distribución equitativa de la carga entre los nodos de la base de datos. Esto significa que los datos deben distribuirse por igual entre los nodos para que ninguno de ellos se cargue de manera desproporcionada.
- Tolerancia a fallas: las particiones deben poder manejar las fallas de los nodos sin afectar la disponibilidad de los datos. Esto significa que la base de datos debe detectar y recuperarse automáticamente de fallas, transferir datos a otros nodos y almacenar datos.
- Escalabilidad horizontal: la fragmentación debe permitir la escalabilidad horizontal de la base de datos, es decir, la capacidad de agregar nuevos nodos según sea necesario para manejar un mayor número de usuarios y transacciones. Coherencia de los datos: la partición debe garantizar la coherencia de los datos entre los nodos. Esto significa que los datos deben replicarse entre nodos de manera coherente y sincronizada para evitar conflictos de datos.
- Rendimiento: las particiones deben mejorar el rendimiento de la base de datos NoSQL. Esto significa que la información debe compartirse de tal manera que las solicitudes puedan procesarse de manera rápida y eficiente.
- Facilidad de uso: las secciones deben ser fáciles de administrar y mantener. Esto significa que debería ser fácil agregar y eliminar nodos, transferir datos entre nodos y realizar tareas de mantenimiento, como copias de seguridad y restauración.
- Seguridad: las particiones deben ser seguras y proteger los datos del acceso no autorizado. Esto significa que cada nodo debe contar con medidas de seguridad como autenticación, autorización y encriptación de datos.

Partición de DB NoSQL

- Después de confirmar el estado del balanceador pasaremos a activar:

```
mongo: > sh.setBalancerState(true)
```



```

s20006| 2023-04-15T23:15:50.058-0500 D1 TRACKING [UserCacheInvalidator] Cmd: N
otSet, TrackingId: 643b76762ef07f860c04668a
s20006| 2023-04-15T23:15:51.237-0500 D1 TRACKING [Uptime-reporter] Cmd: NotSet
, TrackingId: 643b76772ef07f860c04668c
s20006| 2023-04-15T23:15:53.099-0500 I COMMAND [conn12] command admin.$cmd a
ppName: "MongoDB Shell" command: replSetGetStatus { replSetGetStatus: 1.0, for
Shell: 1.0, $clusterTime: { clusterTime: Timestamp(1681618322, 1), signature:
{ hash: BinData(0, 00000000000000000000000000000000), keyId: 0 } }, $d
b: "admin" } numYields:0 reslen:241 protocol:op_msg 0ms
s20006| 2023-04-15T23:15:53.103-0500 D1 NETWORK [ReplicaSetMonitor-TaskExecut
or] Refreshing replica set __unknown_name__-configRS took 0ms
s20006| 2023-04-15T23:15:55.904-0500 D1 SHARDING [shard-registry-reload] Reloa
ding shardRegistry
s20006| 2023-04-15T23:15:55.904-0500 D1 TRACKING [shard-registry-reload] Cmd:
NotSet, TrackingId: 643b767b2ef07f860c046692
s20006| 2023-04-15T23:15:55.905-0500 D1 SHARDING [shard-registry-reload] found
3 shards listed on config server(s) with lastVisibleOpTime: { ts: Timestamp(1
681618551, 1), t: 1 }
s20006| 2023-04-15T23:15:55.905-0500 D1 NETWORK [shard-registry-reload] Start
ed targeter for __unknown_name__-rs0/DESKTOP-RQP70V3:20000
s20006| 2023-04-15T23:15:55.905-0500 D1 NETWORK [shard-registry-reload] Start
ed targeter for __unknown_name__-rs1/DESKTOP-RQP70V3:20001
s20006| 2023-04-15T23:15:55.905-0500 D1 NETWORK [shard-registry-reload] Start
ed targeter for __unknown_name__-rs2/DESKTOP-RQP70V3:20002
s20006| 2023-04-15T23:15:55.915-0500 D1 TRACKING [conn12] Cmd: balancerStart,
TrackingId: 643b767b2ef07f860c046694
c20003| 2023-04-15T23:15:55.916-0500 I SHARDING [conn93] about to log metadat
a event into actionlog: { id: "DESKTOP-RQP70V3:20003-2023-04-15T23:15:55.916-
0500-643b767b2ef07f860c046694", server: "DESKTOP-RQP70V3:20003", shard: "confi
g", clientAddr: "192.168.56.1:62900", time: new Date(168161855916), what: "ba
lancer.start", ns: "", details: {} }
s20006| 2023-04-15T23:15:55.978-0500 I COMMAND [conn12] command admin.$cmd a
ppName: "MongoDB Shell" command: balancerStart { balancerStart: 1.0, maxTimeMS
: 60000.0, lsid: { id: UUID("3394ef49-f6c4-477c-ae0e-a34538a6248f") }, $cluste
rTime: { clusterTime: Timestamp(1681618551, 1), signature: { hash: BinData(0,
00000000000000000000000000000000), keyId: 0 } }, $db: "admin" } numYie
lds:0 reslen:163 protocol:op_msg 63ms
s20006| 2023-04-15T23:15:55.980-0500 I COMMAND [conn12] command admin.$cmd a
ppName: "MongoDB Shell" command: replSetGetStatus { replSetGetStatus: 1.0, for
Shell: 1.0, $clusterTime: { clusterTime: Timestamp(1681618555, 1), signature:
{ hash: BinData(0, 00000000000000000000000000000000), keyId: 0 } }, $d
b: "admin" } numYields:0 reslen:241 protocol:op_msg 0ms

```

```

C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe
mongo> sh.setBalancerState(true)
{
  "ok" : 1,
  "operationTime" : Timestamp(1681618555, 1),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1681618555, 1),
    "signature" : {
      "hash" : BinData(0, "AAAAAAAAAAAAAAAAAAAAA
AAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
mongo>

```

```

C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe
Skate
> shard1DB.trucos.count()
0
> shard1DB.trucos.count()
0
> shard1 = new Mongo("localhost:20000")
connection to localhost:20000
> shard1DB = shard1.getDB("Skate")
Skate
> shard1DB.trucos.count()
0
>

```

- Ahora confirmaremos la distribución en cada uno de los tres nodos:

- Nodo 1

Primero nos conectamos al nodo

```
mongo: > shard1 = new Mongo("localhost:20000")
```

Nos conectamos a la base de datos Skate

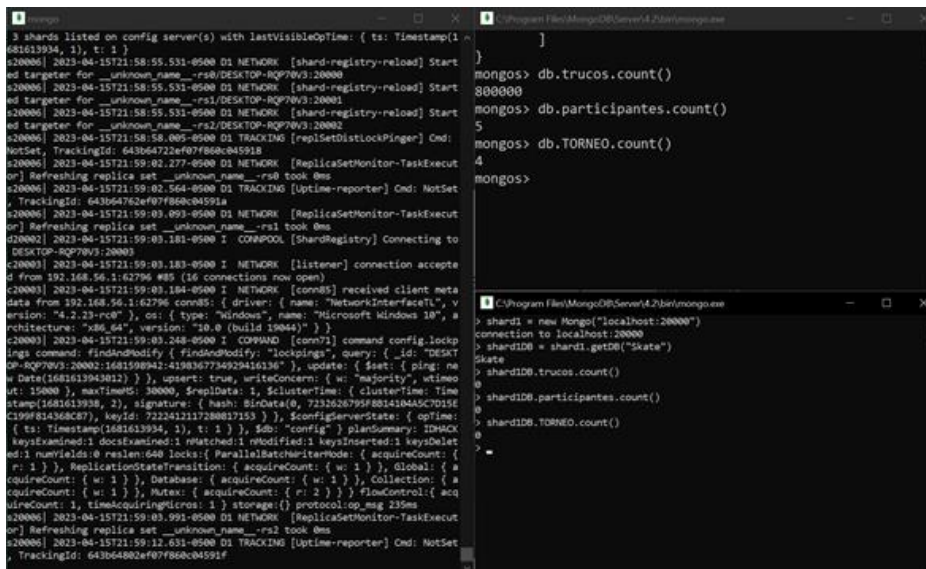
```
mongo: > shard1DB = shard1.getDB("Skate")
```

Revisamos si en este nodo se realizó la inserción en las tres colecciones o no

```
mongo: > shard1DB.trucos.count()
```

```
mongo: > shard1DB.participantes.count()
```

```
mongo: > shard1DB.TORNEO.count()
```



b. Nodo 2

Primero nos conectamos al nodo

```
mongo: > shard2 = new Mongo("localhost:20000")
```

Nos conectamos a la base de datos Skate

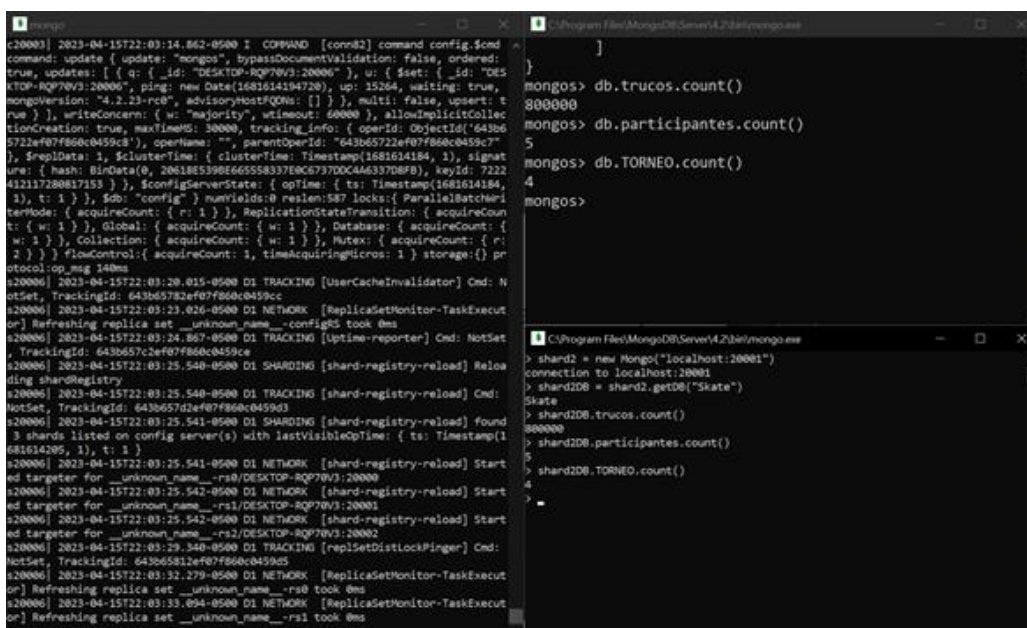
```
mongo: > shard2DB = shard2.getDB("Skate")
```

Revisamos si en este nodo se realizó la inserción en las tres colecciones o no

```
mongo: > shard2DB.trucos.count()
```

```
mongo: > shard2DB.participantes.count()
```

```
mongo: > shard2DB.TORNEO.count()
```



c. Nodo 3

Primero nos conectamos al nodo

```
mongo: > shard3 = new Mongo("localhost:20000")
```

Nos conectamos a la base de datos Skate

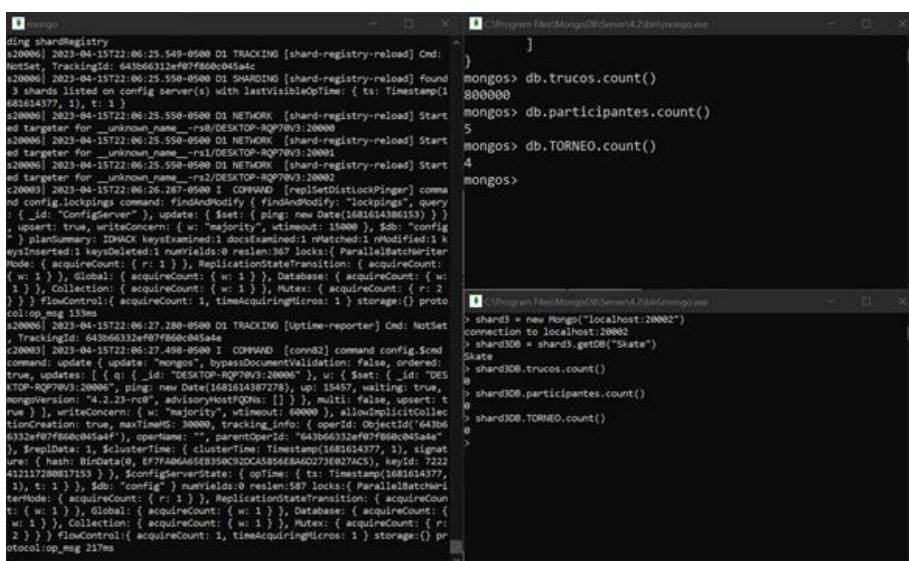
```
mongo: > shard3DB = shard3.getDB("Skate")
```

Revisamos si en este nodo se realizó la inserción en las tres colecciones o no

```
mongo: > shard3DB.trucos.count()
```

```
mongo: > shard3DB.participantes.count()
```

```
mongo: > shard3DB.TORNEO.count()
```



```

mongod
2023-04-15T22:06:25.549-0500 D1 TRACKING [shard-registry-reload] Cmd: NotSet
2023-04-15T22:06:25.550-0500 D1 SHARDING [shard-registry-reload] Found 3 shards listed on config server(s) with lastVisibleOptime: { ts: Timestamp(1681614377, 1), t: 1 }
2023-04-15T22:06:25.550-0500 D1 NETWORK [shard-registry-reload] Start ad targeter for _unknown_name__rs0/DESKTOP-RQP7WV3:20000
2023-04-15T22:06:25.550-0500 D1 NETWORK [shard-registry-reload] Start ad targeter for _unknown_name__rs1/DESKTOP-RQP7WV3:20001
2023-04-15T22:06:25.550-0500 D1 NETWORK [shard-registry-reload] Start ad targeter for _unknown_name__rs2/DESKTOP-RQP7WV3:20002
2023-04-15T22:06:26.287-0500 I COMMAND [replSetDistLockPinger] command config.lockings command: findAndModify { findAndModify: 'lockings', query: { _id: 'ConfigServer' }, update: { $set: { ping: new Date(168161438153) } } }, upsert: true, writeConcern: { w: 'majority', timeout: 15000 }, $db: 'config' }
planSummary: IDMAP keysExamined:1 docsExamined:1 rMatched:1 rModified:1 k keysInserted:1 keysDeleted:1 numFields:0 resLen:367 locks: { ParallelBatcherNode: { acquireCount: { r: 1 }, ReplicationStateTransition: { acquireCount: { w: 1 }, Global: { acquireCount: { w: 1 }, Database: { acquireCount: { w: 1 }, Collection: { acquireCount: { w: 1 }, Mutex: { acquireCount: { r: 2 } } } } } } flowControl: { acquireCount: 1, timeAcquiringMicros: 1 } storage: {} protocol_op_msg 138e
2023-04-15T22:06:27.280-0500 D1 TRACKING [uptime-reporter] Cmd: NotSet
2023-04-15T22:06:27.498-0500 I COMMAND [conn2] command config.$cmd command: update { update: 'mongos', bypassDocumentValidation: false, ordered: true, updates: [ { u: { _id: 'DESKTOP-RQP7WV3:20000' }, u: { $set: { _id: 'DESKTOP-RQP7WV3:20000', ping: new Date(1681614387278), up: 15457, waiting: true, mongoversion: '4.2.23-rc0', advisoryHostQDNs: [ ] } }, multi: false, upsert: true } ], writeConcern: { w: 'majority', timeout: 60000 }, allowImplicitCollectionCreation: true, maxTimeMS: 30000, trackingInfo: { opid: ObjectId('643b643b2e9f7f60b0454e'), operation: 'u', parentOpId: '643b643b2e9f7f60b0454e' }, $preData: 1, $clusterTime: { clusterTime: Timestamp(1681614377, 1), signature: { hash: BinData(0, 8f7f60a6548350c920c45856e8a6d273e027ac5), keyId: 722241211720817353 } }, $configServerState: { optime: { ts: Timestamp(1681614377, 1), t: 1 }, $db: 'config' }, numFields:0 resLen:387 locks: { ParallelBatcherNode: { acquireCount: { r: 1 }, ReplicationStateTransition: { acquireCount: { w: 1 }, Global: { acquireCount: { w: 1 }, Database: { acquireCount: { w: 1 }, Collection: { acquireCount: { w: 1 }, Mutex: { acquireCount: { r: 2 } } } } } } flowControl: { acquireCount: 1, timeAcquiringMicros: 1 } storage: {} protocol_op_msg 217e

C:\Program Files\MongoDB\Server\4.2\bin\mongo.exe
>
>
> shard3 = new Mongo("localhost:20000")
> connection to localhost:20000
> shard3DB = shard3.getDB("Skate")
>
> shard3DB.trucos.count()
800000
> shard3DB.participantes.count()
5
> shard3DB.TORNEO.count()
4
>
>

```

Casos de prueba

Tipo de prueba	Objetivo
Particionamiento	Verificar que se dividió físicamente la información almacenada en la base de datos en 2 shards y a su vez replicar la información de cada parte o shard como se hizo previamente con el esquema centralizado con réplicas
Carga de datos	Validar tiempo de inserción de datos, colección de Equipo sea inferior a un segundo.
Tiempos de respuesta	Validar tiempo de respuesta consulta de datos, alguna de las colecciones de la BD propuesta sea inferior a un segundo.
Validación balanceo de carga de datos en los diferentes nodos.	Validar el funcionamiento del balanceador de carga de datos mediante los diferentes nodos creados, al hacer inserciones masivas de datos en alguna de las colecciones de la BD propuesta.

Script

https://github.com/Idromeromo/Practicas_Ibero_DBAvanzada/tree/particion

Video

<https://youtu.be/DGXplbbupzE>