



Relatório de Projeto de Sistema de Gerenciamento de Estoque para Supermercados

1) Introdução

Este relatório descreve o projeto de um sistema de gerenciamento de estoque para uma cadeia de supermercados com filiais em diferentes cidades. O sistema precisa ser escalável e eficiente, lidando com milhões de registros de produtos, permitindo consultas rápidas e atualizações de inventário. O foco principal é a estratégia de particionamento de dados para garantir desempenho e escalabilidade.

2) Estratégia de Particionamento

Considerando os requisitos apresentados, a estratégia escolhida será a de *particionamento horizontal (Sharding)*, seguem as justificativas para a escolha:

- **Escalabilidade:** Como cada shard pode ser colocado em um servidor diferente, adicionar novas filiais simplesmente envolve adicionar novos shards. Isso permite que o sistema escale horizontalmente com pouca reconfiguração.
 - **Desempenho:** Distribuir os dados entre vários shards reduz a carga em cada servidor individual, melhorando o desempenho geral das consultas e atualizações.
 - **Localidade dos Dados:** Filiais em diferentes cidades provavelmente operam de maneira independente. Particionar os dados por cidade (horizontalmente) significa que a maioria das consultas e atualizações será local a um shard específico, melhorando a eficiência.
 - **Isolamento de Falhas:** Problemas em um shard não afetam outros shards. Isso significa que uma falha em uma filial (cidade) não comprometerá o sistema inteiro.
- Implementação Detalhada

Muitos sistemas não precisam de sharding inicialmente, sendo melhor implementá-lo apenas quando a escalabilidade vertical não é mais suficiente. Portanto, para uma cadeia de supermercados que terá um volume grande de dados, a melhor estratégia é a de particionamento horizontal (sharding). Sem a estratégia

de particionamento, o banco de dados pode rapidamente ficar sobrecarregado, tornando alguns nós ineficientes. O sharding facilita o gerenciamento das informações, garantindo eficiência de desempenho e escalabilidade contínua.

- **Ferramentas utilizadas:** Para realizar a tarefa, foram usadas as seguintes ferramentas: **ChatGPT** para criação de um script de uma base de exemplo, **Python com IDE VSCode** para execução do Script de criação do arquivo json, **Docker** contendo o ambiente dentro de um contêiner e **MongoDB** para leitura e gravação dos dados do database(trabalho_final_bd) e Collection (supermercado). A utilização dos mesmos, foi escolhida visando o desafio e facilidade e eficácia de uso.

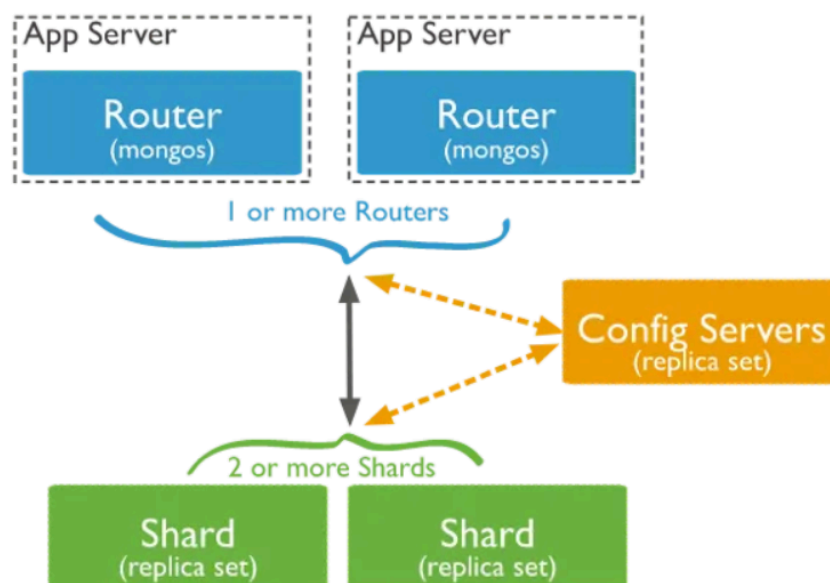
- **Criação do Cluster mongo replicante e particionado:** Para essa tarefa foi usado o método abaixo:

Criação de três tipos de serviços:

Roteadores: responsáveis por receberem as requisições de leitura e escrita e redirecionar para a partição correta.

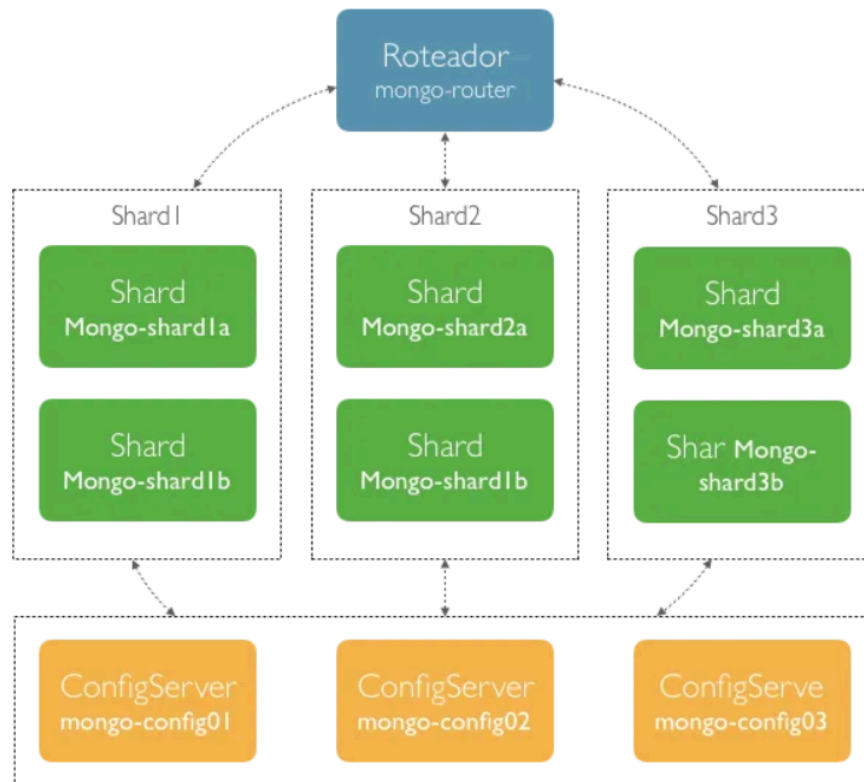
Config Servers: armazenam os metadados das partições (shards). Os metadados incluem a lista de chunks de cada shard e os intervalos que definem os chunks.

Shards: são os nós responsáveis pelo armazenamento dos dados. Cada shard fica responsável por um subconjunto dos dados do banco.



Tipos de serviços do MongoDB

E criação de um cluster com três nós replicantes de configuração (ConfigServer), três shards cada um com uma réplica (totalizando seis nós) e um roteador.



Arquitetura do cluster a ser criado

3) Procedimentos:

a) Criação da rede para a comunicação entre os containers:
















```
C:\Users\Jacson Alves>docker network create mongo-cluster
Error response from daemon: network with name mongo-cluster already exists
```

b) Criação dos containers ConfigServers

```
C:\Users\Jacson Alves>docker run --name mongo-config1 --net mongo-cluster -d mongo mongod --configsvr --replset config-servers --port 27017
994922cc2f9cc077b7a82720cf0512d8936e017effb27eb2c855febee0afa69d

C:\Users\Jacson Alves>docker run --name mongo-config2 --net mongo-cluster -d mongo mongod --configsvr --replset config-servers --port 270172
e15efc77dca66f5caeddecba7edc8e4adc0d46d241aeb0ea8b68706f216fc5

C:\Users\Jacson Alves>docker run --name mongo-config3 --net mongo-cluster -d mongo mongod --configsvr --replset config-servers --port 270172
299f511066d0f3589fa76c5e17618fcdcbd151f8d0ad64b1a410f2390520ab76
```

<input type="checkbox"/>	 mongo-config-1 6fd7c92076ad 	mongo	Running	2.89%	22 hours ago	  
<input type="checkbox"/>	 mongo-config-2 4ecc3dca8db3 	mongo	Running	2.29%	22 hours ago	  
<input type="checkbox"/>	 mongo-config-3 5b7c8fbdf3d9 	mongo	Running	2.05%	22 hours ago	  

c) Inicialização da operação dos servidores de configuração:

```
test> rs.initiate({_id: "config-servers",configsvr: true,version: 1,members:[{_id: 0, host:"mongo-config-1:27017"},{_id: 1, host:"mongo-config-2:27017"},{_id: 2, host:"mongo-config-3:27017"}]})
{ ok: 1 }
```

d) Criação dos Shards em que cada um possui um réplica:

Shards-1:

```
C:\Users\Jacson Alves>docker run --name mongo-shard-1-a --net mongo-cluster -d mongo mongod --shardsvr --replSet shards-1 --port 27017
352eec76bbc72a69df4583669c3870b056a5aeba3efa4d79666ad1086dab24d5

C:\Users\Jacson Alves>docker run --name mongo-shard-1-b --net mongo-cluster -d mongo mongod --shardsvr --replSet shards-1 --port 27017
fdd4c4bdd9bf51f7d0a4ee3c524ce028aab3337e525fe1e4b0bd31676109c7b3
```

Shards-2


























```
C:\Users\Jacson Alves>docker run --name mongo-shard-2-a --net mongo-cluster -d mongo mongod --shardsvr --replSet shards-2 --port 27017
5f5747e3f9127b4092766dd5d9c622ead19e140d1117555f7840e26e608f7c39

C:\Users\Jacson Alves>docker run --name mongo-shard-2-b --net mongo-cluster -d mongo mongod --shardsvr --replSet shards-2 --port 27017
5a9e21bece2375055c728e1f100663a2d54e20c1c6574377e9c2aea9c80aa3fc
```

Shards-3







```
C:\Users\Jacson Alves>docker run --name mongo-shard-3-a --net mongo-cluster -d mongo mongod --shardsvr --replSet shards-3 --port 27017
9a32eb37e0d6098b38faba6fd3b6af6a940b0c56394888bc7f637a55c46639b4

C:\Users\Jacson Alves>docker run --name mongo-shard-3-b --net mongo-cluster -d mongo mongod --shardsvr --replSet shards-3 --port 27017
90a050e181d0f4910d248f269900e70544851ea979122674e58c9b785a0cd36f
```

<input type="checkbox"/>	 mongo-shard-1-a 352eec76bbc7 	mongo	Running	2.57%	22 hours ago			
<input type="checkbox"/>	 mongo-shard-1-b fdd4c4bdd9bf 	mongo	Running	2.91%	22 hours ago			
<input type="checkbox"/>	 mongo-shard-2-a 5f5747e3f912 	mongo	Running	3.1%	22 hours ago			
<input type="checkbox"/>	 mongo-shard-2-b 5a9e21bece23 	mongo	Running	2.23%	22 hours ago			
<input type="checkbox"/>	 mongo-shard-3-a 9a32eb37e0d6 	mongo	Running	2.34%	22 hours ago			
<input type="checkbox"/>	 mongo-shard-3-b 90a050e181d0 	mongo	Running	3.15%	22 hours ago			

e) Inicialização do serviço de roteamento:

```
C:\Users\Jacson Alves>docker run -p 27017:27017 --name mongo-router --net mongo-cluster -d mongo mongos --configdb config-servers/mongo-config-1:27017,mongo-config-2:27017,mongo-config-3:27017 --port 27017 --bind_ip_all
680a7d3740fa27a2571f5cb1d59bc0c5c58cf37ec70c5d6e2d599666e490e434
```

<input type="checkbox"/>	 mongo-router 680a7d3740fa 	mongo	Running	27017:27017 	0.72%	19 hours ago			
--------------------------	---	-----------------------	---------	---	-------	--------------	---	---	---

f) Configuração do roteador para reconhecimento dos shards:

```
C:\Users\Jackson Alves>docker exec -it mongo-router mongosh
Current Mongosh Log ID: 666b6e4a0fbe91cbaa2202d7
Connecting to:      mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+2.2.5
Using MongoDB:      7.0.9
Using Mongosh:       2.2.5

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

-----
The server generated these startup warnings when booting
2024-06-13T22:10:10.993+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----

[direct: mongos] test> sh.addShard("shards-1/mongo-shard-1-a:27017")
{
  shardAdded: 'shards-1',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1718337237, i: 4 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1718337237, i: 4 })
}

[direct: mongos] test> sh.addShard("shards-1/mongo-shard-1-b:27017")
{
  shardAdded: 'shards-1',
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1718337264, i: 2 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1718337264, i: 2 })
}
```

g) Verificação de status do funcionamento do cluster

```
[direct: mongos] trabalho_final_bd> sh.status()
shardingVersion
{ _id: 1, clusterId: ObjectId('666b49c5f25b025cc47f291d') }
---
shards
[
  {
    _id: 'shards-1',
    host: 'shards-1/mongo-shard-1-a:27017,mongo-shard-1-b:27017',
    state: 1,
    topologyTime: Timestamp({ t: 1718337237, i: 1 })
  },
  {
    _id: 'shards-2',
    host: 'shards-2/mongo-shard-2-a:27017,mongo-shard-2-b:27017',
    state: 1,
    topologyTime: Timestamp({ t: 1718337298, i: 2 })
  },
  {
    _id: 'shards-3',
    host: 'shards-3/mongo-shard-3-a:27017,mongo-shard-3-b:27017',
    state: 1,
    topologyTime: Timestamp({ t: 1718337363, i: 2 })
  }
]
---
active mongoses
[ { '7.0.9': 1 } ]
---
autosplit
{ 'Currently enabled': 'yes' }
---
balancer
{
  'Currently enabled': 'yes',
  'Currently running': 'no',
  'Failed balancer rounds in last 5 attempts': 0,
  'Migration Results for the last 24 hours': 'No recent migrations'
}
---
databases
[
  {
    database: { _id: 'config', primary: 'config', partitioned: true },
    collections: {
      'config.system.sessions': {
        shardKey: { _id: 1 },
        unique: false,
        balancing: true,
        chunkMetadata: [ { shard: 'shards-1', nChunks: 1 } ],
```

```

        chunks: [
          { min: { _id: MinKey() }, max: { _id: MaxKey() }, 'on shard': 'shards-1', 'last modified': Timestamp({ t: 1, i: 0 }) }
        ],
        tags: []
      }
    },
  },
  {
    database: {
      _id: 'test',
      primary: 'shards-3',
      partitioned: false,
      version: {
        uuid: UUID('c8f7b14c-b73d-4b04-819b-58a5707f2db8'),
        timestamp: Timestamp({ t: 1718339418, i: 1 }),
        lastMod: 1
      }
    },
    collections: {}
  },
  {
    database: {
      _id: 'trabalho_final_bd',
      primary: 'shards-2',
      partitioned: false,
      version: {
        uuid: UUID('0ab255f9-8026-44b5-866c-4e0cb4b8f123'),
        timestamp: Timestamp({ t: 1718339855, i: 1 }),
        lastMod: 1
      }
    },
    collections: {
      'trabalho_final_bd.supermercado': {
        shardKey: { City: 1 },
        unique: false,
        balancing: true,
        chunkMetadata: [ { shard: 'shards-2', nChunks: 1 } ],
        chunks: [
          { min: { City: MinKey() }, max: { City: MaxKey() }, 'on shard': 'shards-2', 'last modified': Timestamp({ t: 1, i: 0 }) }
        ],
        tags: []
      }
    }
  }
}
]

```

```

  },
  {
    database: {
      _id: 'test',
      primary: 'shards-3',
      partitioned: false,
      version: {
        uuid: UUID('c8f7b14c-b73d-4b04-819b-58a5707f2db8'),
        timestamp: Timestamp({ t: 1718339418, i: 1 }),
        lastMod: 1
      }
    },
    collections: {}
  },
  {
    database: {
      _id: 'trabalho_final_bd',
      primary: 'shards-2',
      partitioned: false,
      version: {
        uuid: UUID('0ab255f9-8026-44b5-866c-4e0cb4b8f123'),
        timestamp: Timestamp({ t: 1718339855, i: 1 }),
        lastMod: 1
      }
    },
    collections: {
      'trabalho_final_bd.supermercado': {
        shardKey: { City: 1 },
        unique: false,
        balancing: true,
        chunkMetadata: [ { shard: 'shards-2', nChunks: 1 } ],
        chunks: [
          { min: { City: MinKey() }, max: { City: MaxKey() }, 'on shard': 'shards-2', 'last modified': Timestamp({ t: 1, i: 0 }) }
        ],
        tags: []
      }
    }
  }
}
]
... { "City": "New York", "Product": "Apple", "Quantity": 100 },
... { "City": "Los Angeles", "Product": "Banana", "Quantity": 150 },
... { "City": "Chicago", "Product": "Orange", "Quantity": 200 }
... ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('666bcaf30f91c8aa2202d8'),
    '1': ObjectId('666bcaf30f91c8aa2202d9'),
    '2': ObjectId('666bcaf30f91c8aa2202da')
  }
}

```

Configuração com status ok.

4) Cenários de testes:

Consulta de documentos:

```
[direct: mongos] trabalho_final_bd> db.supermercado.find({ "City": "New York" }).pretty()
[
  {
    _id: ObjectId('666bcaf30fbe91cbaa2202d8'),
    City: 'New York',
    Product: 'Apple',
    Quantity: 100
  }
]
```

Contagem de documentos: (1934 documentos existentes)

```
[direct: mongos] trabalho_final_bd> db.supermercado.find({ "City": "New York" }).count();
1934
```

Atualização de documentos: (Quantidade atualizada de 100 para 320)

```
[direct: mongos] trabalho_final_bd> db.supermercado.updateOne({ "City": "New York", "Product": "Apple" }, { $set: { "Quantity": 320 } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
[direct: mongos] trabalho_final_bd> db.supermercado.find({ "City": "New York" }).pretty()
[
  {
    _id: ObjectId('666bcaf30fbe91cbaa2202d8'),
    City: 'New York',
    Product: 'Apple',
    Quantity: 320
  }
]
```

Verificação de distribuição de dados:

```
[direct: mongos] trabalho_final_bd> db.adminCommand({ balancerCollectionStatus: "trabalho_final_bd.supermercado" })
{
  chunkSize: 128,
  balancerCompliant: true,
  ok: 1,
  '$clusterTime': {
    clusterTime: Timestamp({ t: 1718341151, i: 1 }),
    signature: {
      hash: Binary.createFromBase64('AAAAAAAAAAAAAAAAAAAAAAAAAAAA=', 0),
      keyId: Long('0')
    }
  },
  operationTime: Timestamp({ t: 1718341151, i: 1 })
}
```


5) Testes de Performance:

- **Inserção em Massa (Medição de Tempo):** Para testar a eficiência do particionamento durante operações de escrita de um grande número de documentos

```
[direct: mongos] trabalho_final_bd> const bulkInsert = () => {  
...   const start = new Date().getTime();  
...   for (let i = 0; i < 10000; i++) {  
...     db.supermercado.insertOne({  
...       "City": ["New York", "Los Angeles", "Chicago", "Houston", "Phoenix"][Math.floor(Math.random() * 5)],  
...       "Product": ["Apple", "Banana", "Orange", "Grape", "Pear"][Math.floor(Math.random() * 5)],  
...       "Quantity": Math.floor(Math.random() * 1000)  
...     });  
...   }  
...   const end = new Date().getTime();  
...   print("Bulk insert time: " + (end - start) + " ms");  
... };  
  
[direct: mongos] trabalho_final_bd>  
  
[direct: mongos] trabalho_final_bd> bulkInsert();  
  
Bulk insert time: 115399 ms ←
```

- **Leitura Simples (Medição de tempo):** Para testar a eficiência do particionamento durante operações de leitura

```
[direct: mongos] trabalho_final_bd> const simpleReadTest = () => {  
...   const start = new Date().getTime();  
...   db.supermercado.find({ "City": "New York" }).count();  
...   const end = new Date().getTime();  
...   print("Simple read time: " + (end - start) + " ms");  
... };  
  
[direct: mongos] trabalho_final_bd>  
  
[direct: mongos] trabalho_final_bd> simpleReadTest();  
Simple read time: 35 ms ←
```

- **Leitura Robusta (Medição de tempo):** Para testar a eficiência durante consultas mais complexas

```
[direct: mongos] trabalho_final_bd> const complexReadTest = () => {  
...   const start = new Date().getTime();  
...   db.supermercado.aggregate([  
...     { $match: { "City": "New York" } },  
...     { $group: { _id: "$Product", totalQuantity: { $sum: "$Quantity" } } }  
...   ]).toArray();  
...   const end = new Date().getTime();  
...   print("Complex read time: " + (end - start) + " ms");  
... };  
  
[direct: mongos] trabalho_final_bd>  
  
[direct: mongos] trabalho_final_bd> complexReadTest();  
Complex read time: 18 ms ←  
  
[direct: mongos] trabalho_final_bd>
```

- **Atualização (Medição de tempo):** Para testar a eficiência durante operações de atualização de documentos

```
[direct: mongos] trabalho_final_bd> const updateTest = () => {  
...   const start = new Date().getTime();  
...   db.supermercado.updateMany(  
...     { "City": "New York" },  
...     { $set: { "Quantity": 500 } }  
...   );  
...   const end = new Date().getTime();  
...   print("Update time: " + (end - start) + " ms");  
... };  
  
[direct: mongos] trabalho_final_bd>  
  
[direct: mongos] trabalho_final_bd> updateTest();  
Update time: 373 ms  
  
[direct: mongos] trabalho_final_bd>
```

- **Delete (Medição de tempo):** Para testar a eficiência durante operações de remoção de documentos

```
[direct: mongos] trabalho_final_bd> const deleteTest = () => {  
...   const start = new Date().getTime();  
...   db.supermercado.deleteMany({ "City": "Phoenix" });  
...   const end = new Date().getTime();  
...   print("Delete time: " + (end - start) + " ms");  
... };  
  
[direct: mongos] trabalho_final_bd>  
  
[direct: mongos] trabalho_final_bd> deleteTest();  
Delete time: 186 ms  
  
[direct: mongos] trabalho_final_bd>
```

6) Conclusão:

A estratégia escolhida se mostrou eficaz, cumprindo todos os requisitos necessários. O uso do MongoDB mostrou ótima performance e com escalabilidade diante da base e testes realizados.