UNICESUMAR - CENTRO UNIVERSITÁRIO DE MARINGÁ
CENTRO DE CIÊNCIAS EXATAS TECNOLÓGICAS E AGRÁRIAS
CURSO DE GRADUAÇÃO EM ENGENHARIA DE SOFTWARE

COMO O SHARDING NO MONGODB PODE IMPACTAR NA ESCALABILIDADE DO BANCO DE DADOS MONGODB

PEDRO BERNARDO SANCHEZ

MARINGÁ – PR
2022

Pedro Bernardo Sanchez

COMO O SHARDING NO MONGODB PODE IMPACTAR NA ESCALABILIDADE DO BANCO DE DADOS MONGODB

Artigo apresentado ao Curso de Graduação em Engenharia de Software da UNICESUMAR – Centro Universitário de Maringá como requisito parcial para a obtenção do título de Bacharel(a) em Engenharia de Software, sob a orientação do Prof. Dr. Aparecido Vilela Júnior.

MARINGÁ – PR

2022

FOLHA DE APROVAÇÃO

PEDRO BERNARDO SANCHEZ

COMO O SHARDING NO MONGODB PODE IMPACTAR NA ESCALABILIDADE DO BANCO DE DADOS MONGODB

Artigo apresentado ao Curso de Graduação em Engenharia de Software da UNICESUMAR – Centro Universitário de Maringá como requisito parcial para a obtenção do título de Bacharel(a) em Engenharia de Software, sob a orientação do Prof. Dr. Aparecido Vilela Júnior.

Aprovado em: ____ de _____ de _____.

BANCA EXAMINADORA

_____

Nome do professor – (Titulação, nome e Instituição)

_____

Nome do professor - (Titulação, nome e Instituição)

_____

Nome do professor - (Titulação, nome e Instituição)

COMO O SHARDING NO MONGODB PODE IMPACTAR NA ESCALABILIDADE DO BANCO DE DADOS MONGODB

Pedro Bernardo Sanchez

RESUMO

Como milhões de dados são gerados todos os dias no mundo inteiro, são necessários bancos de dados cada vez mais desenvolvidos e capacitados para administrar toda essa informação. Por conta disso, surgiram diversas soluções para escalabilidade nos bancos de dados, tanto para vertical, que consiste em aprimorar a capacidade do hardware, quanto para horizontal, que consiste em distribuir o trabalho necessário para realizar operações entre diversos servidores. Um dos principais bancos de dados utilizados na criação de aplicações escaláveis é o MongoDB, um banco de dados não relacional orientado a documentos. Nele é possível implementar a técnica de sharding, que impacta diretamente na performance do banco de dados e também no custo necessário para mantê-lo. O principal objetivo e motivação para esta pesquisa foi apresentar quais os impactos causados na performance e custo ao implementar sharding em um banco de dados MongoDB.

Palavras-chave: NoSQL. Performance. Consulta.

HOW SHARDING CAN IMPACT ON SCALABILITY IN MONGODB DATABASE

ABSTRACT

As millions of data are generated every day all over the world, increasingly integrated and capable databases are needed to manage all this information. It consists of improving the hardware's capacity for horizontal scalability, which consists of distributing the work required to perform operations among several servers. One of the main databases used in the creation of scalable applications is MongoDB, which is a document-oriented non-relational database, and it is possible to implement a sharding technique, which directly impacts the performance of the database and also in the cost necessary to maintain it. The main objective and motivation for this research was to present the impacts caused in performance and cost when implementing sharding in a MongoDB database.

Keywords: NoSQL. Performance. Query.

1 INTRODUÇÃO

A sociedade atual está vivendo em uma era em que grandes volumes de dados são gerados e processados todos os dias. Por conta disso, são necessários bancos de dados cada vez mais robustos e escaláveis para suprir determinadas demandas (ALEXANDRE; CAVIQUE, 2013). Entretanto, existem dois tipos de escalonamento: o vertical, que consiste em aprimorar o hardware dos servidores, e o horizontal, que se constitui em aumentar a quantidade de máquinas para suprir as necessidades exigidas (BRIGHENTI, 2014). Contudo, entre os modelos de escalabilidade, o que mais se destaca em relação ao aumento de desempenho e necessidade de menos recurso monetário para implementação é o modelo de escalabilidade horizontal (JUNIOR, 2017).

As técnicas de escalabilidade horizontal geralmente são implementadas em bancos de dados não relacionais, pois necessitam de maior flexibilidade e tolerância a bloqueios para serem implementadas (FREIRE, 2018).

Uma das principais técnicas para desenvolver a escalabilidade horizontal em bancos de dados é o Sharding, que consiste em distribuir o banco de dados em diversas máquinas, dividindo as informações em diferentes máquinas. A motivação para implementação do Sharding em um banco de dados é aumentar a velocidade de processamento de dados, ou seja, aprimorar a performance nas ações de leitura e escrita de dados (MONGODB, 2022).

A principal motivação para elaboração da pesquisa foi compreender como o banco de dados não relacional MongoDB se comporta em relação à aplicação da técnica de sharding e como consequência identificar os impactos causados na performance do banco de dados e nos custos para mantê-lo.

1.1 Objetivo

O objetivo geral deste trabalho é compreender o funcionamento da implementação de sharding em um banco de dados MongoDB e apresentar os reflexos do sharding na escalabilidade do banco, ou seja, como será refletido na performance e, além disso, o impacto causado nos custos para manter um banco de dados MongoDB sharded cluster.

1.2 Metodologia

Para satisfazer os objetivos específicos da pesquisa serão realizadas pesquisas bibliográficas sobre os modelos relacionais e não relacionais de bancos de dados, pesquisas bibliográficas sobre os tipos bancos não relacionais, escalabilidade vertical e horizontal, pesquisa bibliográfica banco de dados MongoDB e finalmente sobre Sharding no MongoDB. Este é um estudo de natureza básica, com o objetivo de gerar uma pesquisa exploratória de abordagem quantitativa. Para discussão de resultados serão coletadas informações de trabalhos relacionados e coleta de informações de empresa situada na cidade de Maringá, as métricas a serem analisadas são diferenças de tempo para inserção de carga e consulta em banco de dados MongoDB standalone e sharded cluster. Além disso, também foi analisada a diferença de custo para manter o banco de dados MongoDB standalone e sharded cluster na Amazon Web Services.

2 DESENVOLVIMENTO

2.1 Banco de dados

Os bancos de dados e sistemas de bancos de dados transformaram-se em peças de suma importância para o dia a dia da sociedade moderna. Ao longo do dia esbarramos com diversas situações envolvendo um banco de dados. Por exemplo, ao acessar o catálogo de filmes de um cinema, ao realizarmos compras de diversos produtos por meio de uma aplicação web, ou seja, os bancos de dados causam um grande impacto na sociedade e é possível afirmar que desempenham um papel crítico em diversas áreas sociais, sendo elas negócios, engenharia, medicina, direito, educação ciências da informação, entre outras (NAVATHE, 2006).

Um banco de dados é considerado um conjunto de dados, ou seja, uma coleção de dados relacionados. Eles são fatos que podem ser armazenados e possuem sentido subentendido. O conceito apresentado é uma forma genérica de apresentar o conceito de banco de dados, este deve apresentar aspectos do mundo real podendo ser considerado um minimundo ou universo de discurso (UoD). As

alterações que ocorrem dentro do minimundo são retratadas no mundo real (NAVATHE, 2006).

Outra característica de um banco de dados é o fato de que os dados devem ser uma coleção lógica e coerente, ou seja, com algum significado intrínseco, já que um conjunto de dados aleatórios não pode ser interpretado como um banco de dados. Além disso, é possível inferir que ele é desenvolvido, modelado e povoado por dados para suprir uma determinada demanda específica, logo possui um grupo de usuários que interagem com o banco de dados com inserções, atualizações e consultas de dados (NAVATHE, 2006).

Ademais, um banco de dados pode ser de qualquer tamanho e de qualquer profundidade. Por exemplo, o catálogo de filmes de um cinema pode conter mais de um milhão de registros em diferentes categorias. A Internal Revenue Service (IRS), órgão responsável por realizar o gerenciamento dos formulários de impostos preenchidos pelos contribuintes americanos, mantém um banco de dados com uma imensa quantidade de informação, que deve ser organizada e gerenciada para que usuários possam realizar consultas, inserções e alterações em dados necessários.

2.2 Modelos Relacionais

O modelo relacional foi proposto por Edgar Codd em 1970, e nele foi apresentado que os dados organizados de forma relacional permitem uma descrição de uma maneira natural, evitando a necessidade de estrutura adicional para a representação (SOUZA, 2014).

A motivação de Codd na implementação do modelo relacional foi a necessidade de elaborar projetos de banco de dados, recuperar e manipular dados de forma nítida e clara. Edgar classificou essa motivação como Objetivo Independência de Dados (SOUZA, 2014).

No modelo relacional, a representação dos dados é constituída por tabelas com linhas desordenadas, colunas e relação entre tabelas. Uma relação consiste em um esquema, denominado com o nome da relação e de domínio de cada coluna, ou seja, atributo ou campo da relação. Ademais, o domínio também é responsável por restringir os valores que o esquema pode assumir (MACÁRIO; BALDO, 2005).

A figura 1 ilustra como é um esquema com algumas informações de um carro.

Figura 1 – Exemplo de esquema no banco de dados

Carro					
id	modelo	marca	ano	quilometragem	cor
string	string	string	date	float	string

Fonte: Elaborado pelo autor (2022)

Além disso, uma relação também é composta por instâncias de uma relação, ou seja, um conjunto de linhas descritas de acordo com seu próprio esquema, que respeitam o número de atributos definidos. O modelo relacional apenas considera relações as quais correspondem às restrições definidas (MACÁRIO; BALDO, 2005).

A linguagem SQL (*Structured Query Language*), originalmente criada pela IBM para consulta ao seu Sistema-R, transformou-se na linguagem mais utilizada na criação, manipulação e consultas nos sistemas gerenciadores de banco de dados (SGBD) relacionais. Como consequência, ocorreu uma padronização na forma de realizar tais consultas no banco de dados por meio de SGDBs (MACÁRIO; BALDO, 2005).

2.3 Modelos não relacionais

O termo NoSQL foi visto pela primeira vez em 1998, como o nome de um banco de dados não relacional de código aberto, criado por Carol Strozzi. Em meados de 2006, o termo apareceu novamente, citado pelo Google, como um conceito de gerenciamento de megadados, em seu artigo “BigTable: A Distributed Storage System for Structured Data”, o qual destacava ser um banco extremamente escalável e tolerante a falhas, já que os dados eram inseridos de forma indexada e como consequência, as consultas eram mais rápidas (OLIVEIRA, 2014).

O principal objetivo do movimento NoSQL é criar uma forma de resolver problemas de escalabilidade dos bancos de dados tradicionais, já que o custo para tal ação demanda muito recurso monetário, além da complexidade.

A escalabilidade em banco de dados tradicionais acaba sendo feita de forma vertical, ou seja, melhorando recursos de um único computador, que por si só executa todos os programas acessando o banco. Já a escalabilidade em banco de dados não relacionais tende a ser horizontal, que necessita de muito menos

recursos para ser implementada, já que várias máquinas mais simples podem ser responsáveis por manter o banco de dados (MOURA; CASANOVA,1999).

Em 2005 foi lançado um banco de dados chamado CouchDB, que utilizava JSON (*JavaScript Object Notation*) para armazenar os dados. Esse formato é extremamente leve para operações de dados computacionais.

A empresa Amazon publicou em 2007 um artigo chamado “Dynamo: Amazon 's Highly Available Key-value Store”. Dynamo é um banco de dados não relacional de alta disponibilidade com armazenamento de dados baseado em chave-valor nos servidores.

A empresa 10Gen, em 2009, publicou o banco de dados não relacional MongoDB, uma aplicação de alta performance, sem esquemas e orientados a documentos. Utiliza JSON para armazenamento de dados, muito semelhante ao CouchDB, além disso foi escrito na linguagem C++.

2.4 Classificações de modelos não relacionais

2.4.1 Chave-valor

Sistema que armazena valores indexados para recuperação por chaves. Podem armazenar dados estruturados ou não estruturados. Um grande exemplo de banco de dados não relacional baseado em chave-valor é o SimpleDB da Amazon (LEAVITT, 2010).

2.4.2 Orientado a colunas

Diferentemente dos bancos de dados relacionais tradicionais, que armazenam conjuntos de informações em tabelas com campos e tamanhos uniformes, o banco de dados orientado a colunas contêm uma coluna extensível que possibilita uma flexibilização maior na forma de estruturar os dados. Um exemplo de banco de dados orientado a colunas é o Cassandra, criado pelo Facebook (LEAVITT, 2010).

Baseado em documentos: esses bancos de dados armazenam e estruturam as informações em coleções de documentos. Diferentemente de tabelas estruturadas, os campos não precisam ser uniformes em relação ao tamanho e tipo de valor, sendo possível adicionar qualquer número de campos de qualquer

tamanho para um documento. Um exemplo de banco de dados baseado em documentos é o próprio MongoDB, criado pela empresa 10gen (LEAVITT, 2010).

2.4.3 Orientado a grafos

O modelo orientado a grafos é adequado para estruturar relacionamentos complexos entre diversos tipos de entidades. A motivação para esse modelo de banco de dados é a representação de entidades em vértices e seus relacionamentos como arestas (MOREIRA, 2016).

Um banco de dados orientados a grafos é uma plataforma em que é possível criar e manipular dados em forma de grafos, ou seja, contém nós, arestas e propriedades. Esse modelo de banco de dados é muito adequado para descobrir padrões em determinado sistema, como nas redes sociais (ORACLE, 2022).

Um exemplo de banco de dados orientado a grafos é o Neo4j, criado pela Neo4j, INC, em 2007.

2.5 Vantagens do modelo não relacional

Os banco de dados NoSQL geralmente processam dados mais rapidamente que relacionais, já que normalmente seus modelos de dados são mais simples. Logo, o ganho de performance em relação ao modelo relacional é superior (LÓSCIO; OLIVEIRA; PONTES, 2011).

Outra vantagem é a escalabilidade horizontal. Sendo assim, é possível criar aplicações altamente escaláveis com muito menos recurso monetário, além disso outra vantagem é o fato de o modelo não relacional possuir esquema flexível, o que contribui para a escalabilidade e disponibilidade, porém a ausência de esquemas não garante a integridade dos dados (LÓSCIO; OLIVEIRA; PONTES, 2011).

Além disso, uma das principais vantagens do modelo não relacional é o suporte nativo a replicação de dados, uma forma de prover escalabilidade através da replicação, pois a replicação gera como consequência a diminuição do tempo gasto para recuperar informações, ou seja, realizar consultas (LÓSCIO; OLIVEIRA; PONTES, 2011).

2.6 Desvantagens do modelo não relacional

O desenvolvimento de banco de dados relacionais deve atender os requisitos básicos estipulados pelo padrão ACID, que são: atomicidade, consistência, isolamento e durabilidade. No padrão NoSQL, Eric Brewer propôs o teorema de CAP, o qual consiste em consistência dos dados, disponibilidade e tolerância a particionamento. Entretanto, um sistema facilmente escalável só é possível de garantia as duas das três características no teorema de CAP, pois elas podem se anular. Dessa forma, é possível concluir que ao obter disponibilidade e tolerância ao particionamento é impossível ter consistência, já que o usuário pode realizar operações de leitura e escrita a qualquer momento (ANICETO; FREIRE, 2014).

Uma desvantagem no modelo não relacional é o fato de necessitar de “manual query programming”, que pode ser rápida para tarefas simples, porém demorada para tarefas mais complexas, já que consultas complexas tendem a ser mais difíceis de programar em bancos de dados não relacionais (LEAVITT, 2010).

Além disso, outro fator que afeta negativamente o modelo não relacional é a confiabilidade, pois enquanto bancos de dados relacionais atendem ao padrão ACID, o modelo não relacional não oferece nativamente nenhum grau de confiabilidade (LEAVITT, 2010).

O desconhecimento da tecnologia pode ser considerada uma desvantagem, pois a maioria das organizações não está familiarizada com o modelo não relacional, logo não cogitam a escolha de um modelo não relacional para solucionar determinado problema.

2.7 MongoDB

A empresa MongoDB foi fundada em 2007, por Dwight Merriman, Eliot Horowitz e Kevin Ryan. A necessidade da criação do MongoDB se deu quando a empresa de publicidade na internet DoubleClick propagava cerca de 40.000 anúncios por segundo e muitas das vezes passava por problemas de escalabilidade e agilidade (MONGODB, 2022).

O nome MongoDB significa gigantesco e o princípio mais importante nesse banco de dados é o fato de que um só tamanho não é ideal para todos. O MongoDB é um banco de dados não relacional e orientado a documentos. A motivação da

equipe do MongoDB foi de criar um banco de dados extremamente rápido, com baixa curva de aprendizagem, e mais importante, o fato de ser altamente escalável (HOWS; MEMBREY; PLUGGE, 2019).

O principal motivo do MongoDB ser um banco escalável é o fato de que é possível dividir o sistema em diversos servidores, considerado um problema para a maioria dos bancos de dados relacionais tradicionais. Por exemplo, nem todos os bancos são capazes de executar o mesmo banco de dados em diferentes servidores, ou seja, em diferentes máquinas, simultaneamente (HOWS; MEMBREY; PLUGGE, 2019).

A técnica de executar um mesmo banco de dados em diferentes servidores gera como consequência uma escalabilidade horizontal, ou seja, garante um aprimoramento da performance do banco de dados para ações de leitura e escrita de dados com menos recurso monetário.

O MongoDB utiliza o formato BSON (binary JSON) para armazenar em forma de documentos em coleções. O BSON é extremamente semelhante ao JSON, porém conta com alguns recursos os quais tornam o MongoDB mais rápido. Tanto o JSON quanto BSON são formatos capazes de armazenar estruturas de dados complexos, em formato de texto legíveis para seres humanos (HOWS; MEMBREY; PLUGGE, 2019).

A figura 2 ilustra um exemplo de como os dados são armazenados no MongoDB.

Figura 2 – Exemplo de documento de carro armazenado no MongoDB

Fonte: Elaborada pelo Autor (2022)

2.8 Escalabilidade

Escalabilidade representa o crescimento do sistema, pode ser relacionado às funcionalidades e também referente às demandas, por exemplo: a quantidade de usuários aumenta e consequentemente são aumentadas as ações de inserções e leituras no banco de dados (LÓSCIO; OLIVEIRA; PONTES, 2011).

Os bancos de dados relacionais foram arquitetados para serem executados em apenas um servidor, ou seja, para serem instanciados em uma única máquina, e para escalar, isto é, aprimorar o servidor, faz-se necessário adquirir uma máquina melhor ou realizar melhorias nos componentes (POLITOWSKI; MARAN, 2014).

A escalabilidade vertical é um técnica economicamente inviável, pois o recurso monetário para aprimorar componentes é muito alto e também limitado, já que em determinado momento o hardware não pode ser melhorado.

De acordo com a demanda de ações de escrita e leitura de informações no banco de dados, aumenta a necessidade de escalabilidade e melhoria do desempenho, em que a escalabilidade horizontal realiza o aumento no número de máquinas disponíveis para armazenamento e processamento de dados. A escalabilidade horizontal é uma solução monetariamente mais viável, entretanto é necessário que vários processos sejam criados para executar uma tarefa, o que dificulta o uso de banco de dados relacionais tradicionais (LÓSCIO; OLIVEIRA; PONTES, 2011).

A inexistência de bloqueio nos bancos de dados NoSQL permite a escalabilidade horizontal, uma vez que a tecnologia se torna adequada para solucionar problemas de administração de enormes volumes de dados com crescimento exponencial (LÓSCIO; OLIVEIRA; PONTES, 2011).

Uma das principais técnicas para alcançar a escalabilidade horizontal é o Sharding, que consiste em dividir os servidores que executam o banco de dados em múltiplas máquinas a fim de reduzir o custo de processamento para recuperação e armazenamento de dados (SOUZA JUNIOR, 2017).

2.9 SHARDING

O sharding é um método para dividir dados entre diversos servidores. Tal técnica é utilizada no MongoDB para implementar soluções com grande volume de tráfego de informações. A fragmentação do banco de dados utilizando o sharding usa dos princípios de escalabilidade horizontal. Para implementar o Sharding no MongoDB faz-se necessário o uso de três componentes principais: shard, mongos e servidores de configuração (MONGODB, 2022).

O shard é considerado um subconjunto dos dados fragmentados, ou seja, cada shard torna-se uma máquina que executa um banco de dados mongodb e nela são armazenadas informações pertencentes àquele subconjunto específico (MONGODB, 2022).

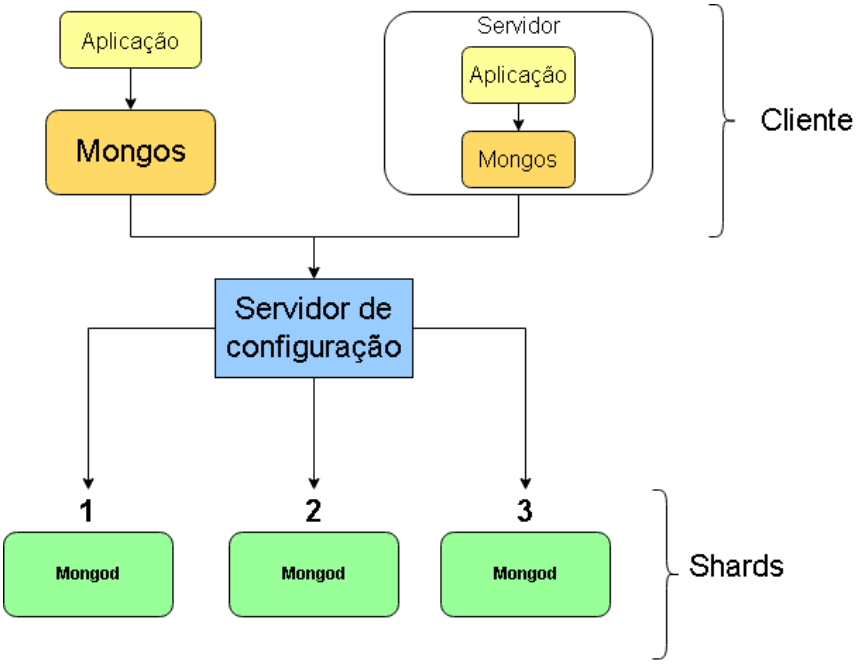
Cada banco de dados em um cluster fragmentado, isto é, na máquina em que está executando o banco de dados referente a determinado shard contém todas as coleções que armazenam as informações direcionadas àquele específico shard.

Os mongos são como interfaces que realizam consultas ou escritas em determinado shard em específico. Por exemplo, a consulta de uma informação que está alocada em um shard é feita a partir dos mongos (MONGODB, 2022).

Os mongos utilizam metadados para redirecionar operações de inserção e leitura de dados. Essas operações são executadas pelas aplicações por meio de uma instância mongod em que o próprio servidor da aplicação realiza a conexão. Também é possível executar instâncias mongod em servidores dedicados, ou seja, separados da aplicação.

Os servidores de configuração têm a função de armazenar metadados de um shard fragmentado, ou seja, reflete o estado e a organização de todos os dados e componentes no cluster fragmentado. Além disso, os servidores de configuração também realizam o armazenamento de configurações de autenticação para funções de controle de acesso para o cluster (MONGODB, 2022).

Figura 3 – Topologia Sharding MongoDB



Fonte: Adaptado de MongoDB Inc (2022)

Uma das principais vantagens em aplicar o sharding no MongoDB é a possibilidade de criar aplicações altamente escaláveis com menos recurso monetário, comparado com bancos de dados relacionais tradicionais. Além disso, a implementação do sharding consequentemente aprimora a performance na execução de ações de leitura e escrita no banco de dados. Logo, o tempo de resposta na execução de tais tarefas diminui consideravelmente (FERREIRA; RIGHI, 2022).

3 RESULTADOS E DISCUSSÃO

3.1 Trabalhos relacionados

As métricas utilizadas para análise de dados foram velocidade na inserção de documentos no teste de carga, tempo necessário para realizar consultas, e diferença de recurso monetário necessário para manter o banco de dados MongoDB standalone e sharded cluster.

Em uma pesquisa realizada por Dourado, Pires, Holanda, Ribeiro e Carvalho comparando o desempenho entre a arquitetura de banco de dados MongoDB

standalone com a arquitetura sharded cluster, num contexto de análise de grande volume de dados provenientes de dados abertos sobre o pagamento da Bolsa Família de 2015, disponibilizados no Portal da Transparência do Ministério da Transparência, Monitoramento e Controladoria-Geral da União.

Os dados disponibilizados pelo Portal da Transparência continham cerca de 14 milhões de registros e foram distribuídos entre sete shards, sendo três máquinas com sistema operacional Ubuntu na versão 14.04 LTS, e as quatro máquinas restantes utilizavam o sistema operacional Windows 7. Além disso, foram distribuídos cerca de 23 milhões de documentos para cada shard.

Tabela 1 – Distribuição de documentos entre shards

Shard	*S.O.*	*Qtd.*	*Desvio*
SHARD000	Windows 7	23.823.907	-0,015%
SHARD001	Windows 7	23.832.189	0,020%
SHARD002	Windows 7	3.824.515	-0,012%
SHARD003	Windows 7	23.825.117	-0,010%
SHARD004	Ubuntu 14 04 LTS	23.835.430	0,034%
SHARD005	Ubuntu 14 04 LTS	23.823.041	-0,018%
SHARD006	Ubuntu 14 04 LTS	23.827.518	0,001%
Total	-	166.791.717	-
Média	-	23.827.388	-

Fonte: (Dourado, Pires, Holanda, Ribeiro e Carvalho, 2017)

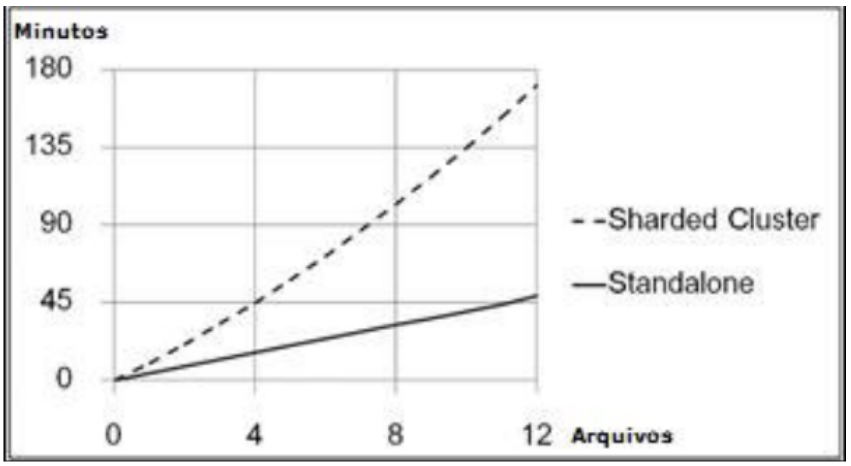
3.1.1 Tempo de carga

Na pesquisa mencionada, o banco de dados standalone provou ser quatro vezes mais rápido na inserção de documentos no teste de carga, sendo que a frequência de inserção de documentos foi de 56.731 documentos/segundo, enquanto o sharded cluster teve uma frequência de 16.256 documentos/segundo.

O resultado apresentado no tempo de carga foi devido ao fato de que enquanto estão sendo inseridos dados no sharded cluster, os próprios shards realizam a distribuição parcialmente igualitária entre eles.

O gráfico 1 apresenta o comparativo entre o tempo de carga entre os dois modelos analisados.

Gráfico 1 – Tempo de carga em relação ao número de documentos inseridos

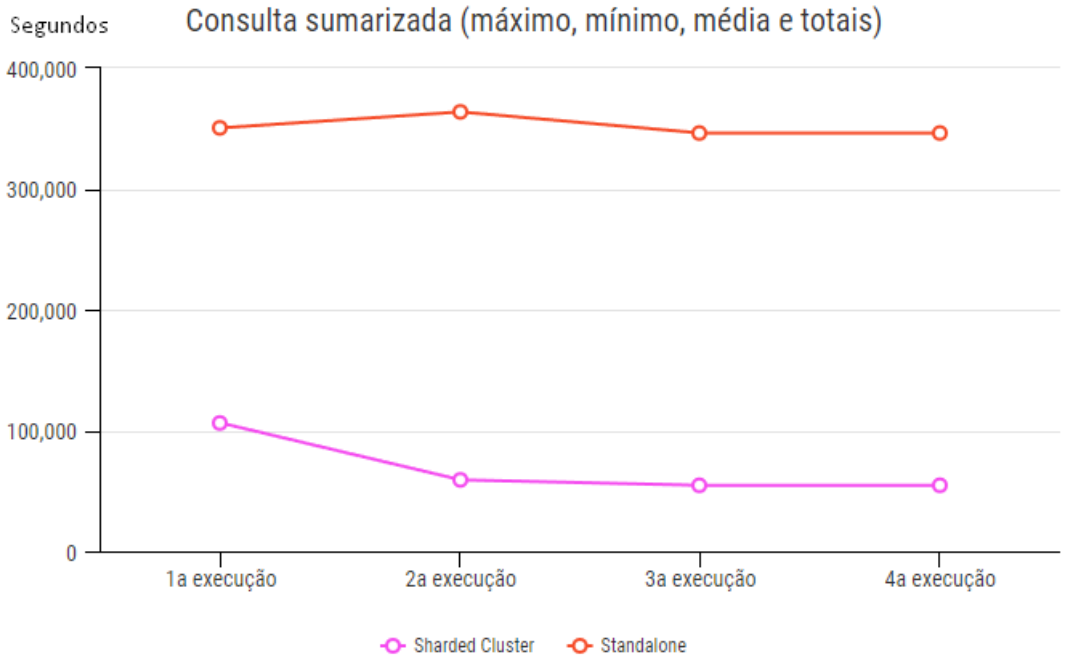


Fonte: (Dourado, Pires, Holanda, Ribeiro e Carvalho, 2017)

3.1.2 Desempenho em consultas

Em relação às consultas foi observado uma diferença significativa de desempenho por parte do modelo sharded cluster. O modelo apresentou ser de 8 a 6 vezes mais eficiente que o modelo standalone. É possível definir que o desempenho é proporcional à quantidade de shards instanciados.

Gráfico 2 – Comparativo de desempenho em consultas entre sharded cluster e standalone

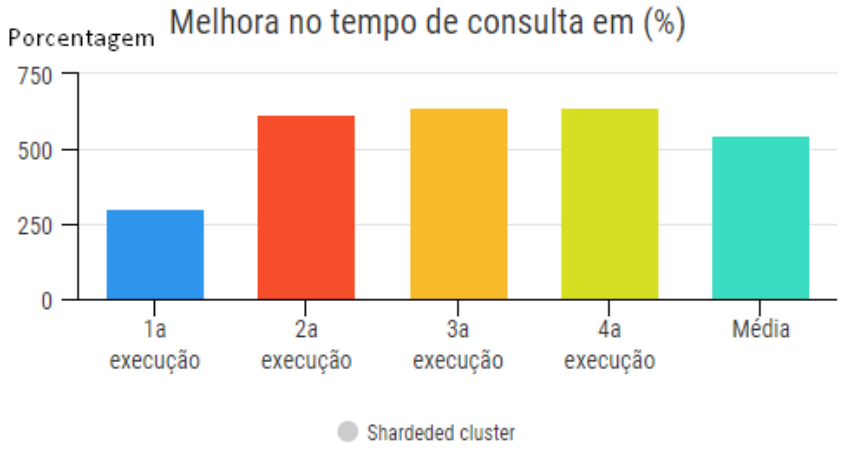


Fonte: Adaptado de Análise de Desempenho do MongoDB (2022)

O tempo de consulta no modelo sharded cluster provou ser consideravelmente mais eficiente, pois obteve-se uma média em percentual de melhora de 500% a 600% no tempo de consulta.

O gráfico 3 ilustra a relação entre execução e a melhora em porcentagem.

Gráfico 3 – Representação da melhora no tempo de consulta



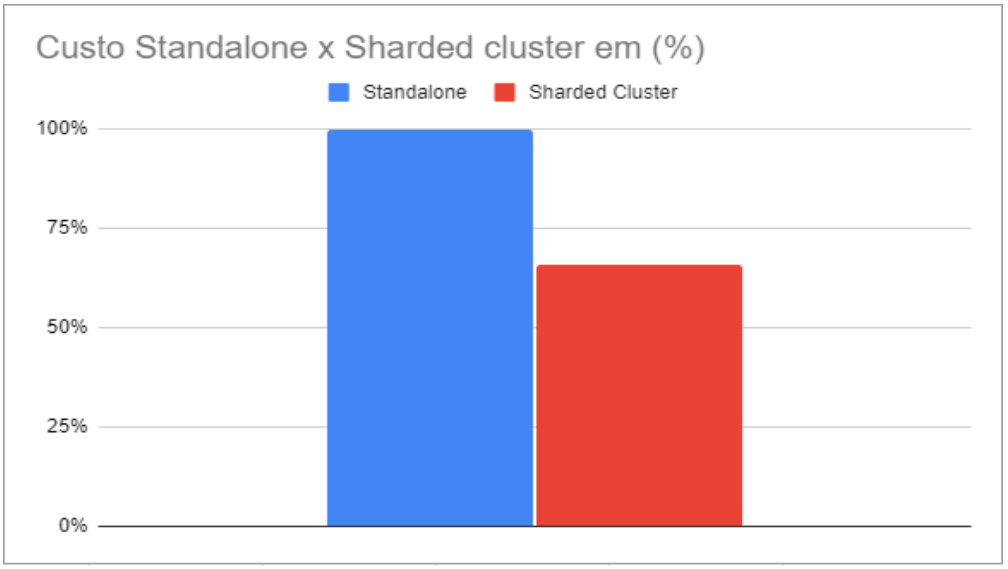
Fonte: Elaborado pelo autor (2022)

3.2 Recurso Monetário

Uma empresa da área tecnologia da informação, situada na cidade de Maringá/PR, disponibilizou a diferença de custo mensal de uma máquina que executava o banco de dados MongoDB padrão (standalone) e passou a executar o banco de dados MongoDB com sharding (sharded cluster). O banco de dados contava com três shards e um servidor de configuração e os mongos são executados nos servidores das próprias aplicações. Também é importante ressaltar que os servidores estão hospedados na Amazon Web Services (AWS).

A diminuição de custo mensal do banco de dados foi de cerca de 33% referente ao total de custo da máquina que executava MongoDB standalone.

Gráfico 4 – Diferença de custo entre MongoDB standalone e sharded cluster



Fonte: Elaborado pelo autor (2022)

4 CONCLUSÃO

Ao verificar os resultados dos testes apresentados, é possível concluir que a implementação do sharding no banco de dados MongoDB é uma solução válida para escalabilidade horizontal, já que se observa ganhos de performance, ou seja, redução no tempo para realização de consultas. Além disso, também foi constatado redução nos custos para manter um banco de dados. Também é importante salientar que bancos de dados não relacionais geralmente são as opções mais adequadas para escalar de forma horizontal, já que grande parte das técnicas de escalabilidade horizontal exigem flexibilidade e tolerância a bloqueios por parte do banco de dados.

Foi averiguado que, apesar de extremamente flexíveis e escaláveis, os bancos de dados não relacionais não são capazes de solucionar todos os problemas e, sim, que existem opções de modelos de bancos dados adequados para determinadas situações.

REFERÊNCIAS

ANICETO, Rodrigo Cardoso; XAVIER, Renê Freire. **Um Estudo Sobre a Utilização do Banco de Dados NoSQL Cassandra em Dados Biológicos**. 2014. Monografia (Bacharelado em Ciência da Computação) – Universidade de Brasília, Brasília, 2014.

CASANOVA, M. A.; MOURA, C.M.O. Sistemas de Gerência de Bancos de Dados Distribuídos São Factíveis?. *In*: XVII Congresso Nacional de Informática, 1984, Rio de Janeiro, RJ, Brasil. **Anais do XVII Congresso Nacional de Informática**. Rio de Janeiro: SUCESU, 1984.

FERREIRA, Daniel Lopes; DA ROSA RIGHI, Rodrigo. Sharding para Acesso Rápido a Dados Móveis e Distribuídos: Uma Aplicação para Smart Cities com Fog Computing. *In*: **Anais da XXII Escola Regional de Alto Desempenho da Região Sul**. SBC, 2022. p. 41-44.

HOWS, Davi; MEMBREY, Peter; PLUGGE, Eelco. Introdução ao MongoDB. **Eduem**, 2019, v. 1, p. 15-18.

JUNIOR, João Bosco de Sousa. **MySQL Cluster Admin**: Uma Ferramenta para Configuração, Gerenciamento e Monitoramento de SGBD NoSQL Fragmentados e Replicados. 2017. Dissertação de Pós-graduação. (Pós-Graduação em Ciência da Computação) - Centro de Informática da Universidade Federal de Pernambuco, Recife, 2017.

LEAVITT, Neal. Will NoSQL Databases Live Up to Their Promise?. **IEEE Computer Society**, v. 43, n. 2, p. 1-3, 2010.

LÓSCIO, Bernadette Farias; OLIVEIRA, Hélio Rodrigues; PONTES, Jonas César de Sousa. **NoSQL no desenvolvimento de aplicações Web colaborativas.** Maringá: Eduem, 2022.

MONGODB. MongoDB Sharding. **MongoDB Inc**, c2022. Disponível em: <https://www.mongodb.com/docs/manual/sharding/> Acesso em: 07 nov. 2022.

MOREIRA, Gustavo Henrique. **Um Modelo de Visualização de Dados Utilizando Banco de Dados Orientado a Grafo Suportado por Big Data.** 2016. Dissertação de Mestrado (Mestre em Engenharia Elétrica) – Universidade de Brasília Faculdade de Tecnologia Departamento de Engenharia Elétrica de Brasília, DF, 2016.

NAVATHE, Ramez Elmasri Shamkant B. **Sistemas de Banco de Dados.** Maringá: Eduem, 2019.

NETO, Aloísio Dourado et al. MongoDB performance analysis: A comparative study between stand-alone and sharded cluster deployments with open data from Brazilian Bolsa familia program. *In*: **2017 12th Iberian Conference on Information Systems and Technologies (CISTI)**. IEEE, 2017. p. 1-6.

OLIVEIRA, Samuel Silva. Bancos de Dados Não-relacionais: Um Novo Paradigma Para Armazenamento de Dados em Sistemas de Ensino Colaborativo. **Revista da Escola de Administração Pública do Amapá**, v. 2, n. 1, p. 4-11, 2014.

ORACLE. Banco de dados de grafos definido. **Oracle Inc**, 2022. Disponível em: <https://www.oracle.com/br/autonomous-database/what-is-graph-database>. Acesso em: 09 nov. 2022.

POLITOWSKI, Cristiano; MARAN, Vinıcius. Comparaç ao de Performance entre PostgreSQL e MongoDB. **X Escola Regional de Banco de Dados, SBC**, p. 1-10, 2014.

RODRIGUES, Joana Micaela da Silva. **Análise de desempenho do MongoDB em cenários de elevada distribuição e partição de dados**. 2018. Dissertação (Mestrado) – Instituto Universitário de Lisboa, ISCTE-IUL, Lisboa, 2018. Disponível em:<https://repositorio.iscte-iul.pt/bitstream/10071/19743/4/master_joana_silva_rodrigues.pdf>. Acesso em: 19 out. 2022.

SOUZA, Odécio. **Edgar Frank Codd e o Banco de Dados Relacional**: Uma contribuição para a História da Computação. 2014. Programa de Estudos Pós-Graduados em História da Ciência (Mestre em História da Ciência) - Pontifícia Universidade Católica de São Paulo PUC-SP, São Paulo, 2014.