REPLICAÇÃO DE BANCO DE DADOS: CLUSTER EM AMBIENTE NA NUVEM COM MONGODB

Victor Aparecido Ribeiro[1]

Vinicius Alcará da Silva[2]

Luis Alexandre da Silva[3]

Resumo

Este trabalho visa demonstrar a utilização de uma rede de cluster não relacional, estruturada sobre o banco MongoDB e disponibilizada em um ambiente hospedado na nuvem, com o intuito do estabelecimento de uma alta performance e disponibilidade para as informações armazenadas nessa estrutura. É uma alternativa viável e eficaz, para empresas de pequeno, médio e grande porte, no combate de falhas que podem comprometer desde a execução de rotinas internas, bem como a interação do usuário com os sistemas. Concluiu-se, com os experimentos realizados, que a utilização de uma infraestrutura externa, torna possível a construção de um sistema de armazenamento de banco de dados não relacional baseado em documentos de forma eficiente, íntegra e sem um investimento financeiro demasiado.

Palavras-chave: Alta disponibilidade em clusters. Banco de dados não relacional. MongoDB. Replicação. Tolerância a falhas em bancos de dados.

Abstract

This paper aims at demonstrating the use of a non-relational cluster network, structured on the MongoDB database and made available in a cloud hosted environment, in order to establish high performance and availability to the information stored in this structure. It is a viable and effective alternative, for small, medium and large companies, in combating failures that can compromise the execution of internal routines, as well as the user's interaction with the system. It was concluded, with the experiments carried out, that the use of an external infrastructure, makes it possible to build a document based non-relational database storage system in an efficient, integral way and without much financial investment.

Keywords: *Fault tolerance in databases. High availability in clusters. MongoDB. Non-relational database.*

1 Introdução

[1] Graduando em Banco de Dados pela Fatec Bauru/SP. Endereço eletrônico: victor.ribeiro7@fatec.sp.gov.br
[2] Graduando em Banco de Dados pela Fatec Bauru/SP. Endereço eletrônico: vinicius.silva308@fatec.sp.gov.br
[3] Docente na Fatec Bauru /SP. Endereço eletrônico: luis.silva51@fatec.sp.gov.br

Com o aumento exponencial do tráfego de dados nos mais variados processos e sistemas online, a necessidade de manter uma aplicação com uma alta disponibilidade e com a garantia de segurança e integridade das informações armazenadas nos bancos de dados se torna um importante diferencial para os fornecedores de hospedagens e softwares. Por mais que os ambientes destinados a estas tarefas procurem adotar rotinas de segurança e adequar seu poder de processamento, falhas podem ocorrer. Para evitar que as consequências de falhas operacionais causem prejuízos financeiros e estruturais, a utilização de bancos de dados não relacionais orientados à documentos tem ganhado força no mercado. De acordo com o site voltado para coleta e apresentação de informações Sistema de Gerenciamento de Banco de Dados (SGBD) DB *Engines*. (Db-Engines1, 2017), o SGBD não relacional mais popular atualmente é o MongoDB, um banco de dados orientado à documentos utilizados por diversas empresas como Facebook, Cisco, EA Games, Ebay, Adobe entre outras.

O crescimento de recursos e volume de informações trafegadas entre sistemas cria a necessidade de se possuir uma estrutura sólida que garanta a máxima capacidade de resguardo de informações e disponibilidade de sistemas. Uma demanda que cresce cada vez mais no mercado da tecnologia da informação. Assim, novas técnicas e tecnologias são adotadas para suprir esta demanda. Segundo Sadalage e Fowler (2013) os entusiastas de bancos *Not Only* SQL (NoSQL) afirmam que determinados sistemas podem ser desenvolvidos mais facilmente, ser mais escaláveis e possuir maior performance.

Outro recurso para se alcançar um nível otimizado de alta disponibilidade é a utilização da clusterização em ambientes responsáveis pela hospedagem de bases de dados com uma grande volumetria armazenada e alta taxa de requisições. Para Tanembaum (2006), clusters são definidos como “sistemas distribuídos onde os processadores não compartilham os mesmos barramentos”. Em uma visão diferente, Stallings (2010) conceitua que clusters são como um grupo de computadores completos interconectados trabalhando juntos, como um recurso computacional unificado que pode criar a ilusão de ser uma única máquina. Esta prática consiste na criação de várias instâncias que serão responsáveis por executar a replicação dos dados existentes na base em questão. A função contribui significativamente para a diminuição dos efeitos colaterais decorrentes de uma falha na estrutura, prevenindo perdas de bases de dados inteiras ou parciais. Desta forma, a clusterização pode

ser uma solução, assim, neste caso, dois ou mais servidores são capazes de trabalhar juntos sob um controle de um sistema operacional, com baixa perda de performance.

A alta disponibilidade na área da computação pode ser obtida pela clusterização, que tem como uma de suas características a utilização de vários hardwares para replicação de dados e sua função é evitar a perda parcial ou total de informações, mantendo seus serviços disponíveis por maior tempo possível no sistema. Para Pitanga (2003), um cluster é um sistema que compreende dois ou mais computadores, ligados em rede, na qual trabalham em conjunto para executar aplicações ou realizar outras tarefas, de tal forma para que os usuários que os utilizam tenham a impressão que somente um único sistema responde para eles, criando assim uma ilusão de um recurso. Segundo Reis *et al*. (2011), clusters geralmente são utilizados para processar conteúdos críticos, garantir a disponibilidade de serviços e manter funcionando o sistema com tolerância às falhas e sem interrupções. Para empresas de *e-commerce*, por exemplo, a indisponibilidade de seu sistema, mesmo que por um minuto, pode alavancar grandes prejuízos financeiros, físicos ou morais à organização, podendo até mesmo levar à falência (NEWS.COMSCHOOL, 2015). Principalmente em datas comemorativas, onde ocorre um maior tráfego de dados simultâneos. No ano de 2008, 90 minutos off-line causaram à Amazon um prejuízo de US$ 2,79 milhões, segundo o site Globo.com (G1.COM, 2008). Já em 2013, o CanalTech revelou que para os varejistas um minuto em que seus sistemas fiquem fora do ar pode levar US$ 8 mil em prejuízos durante a Black Friday (CANALTECH.COM, 2013). A partir desta situação é possível notar que o banco de dados é uma entidade primordial para o armazenamento das informações e deve ser implantado sobre uma estrutura sólida e confiável, para que o usuário final tenha sua necessidade atendida.

O objetivo deste trabalho é contribuir com vias que assegurem a disponibilidade das informações armazenadas em instâncias do banco não relacional MongoDB. Para tal, serão aplicados procedimentos de replicação de dados em serviços de hospedagem na nuvem, como a própria plataforma de *Cloud* MongoDB Atlas, buscando garantir um nível satisfatório e performático das replicações e mantendo a estabilidade e disponibilidade das instâncias. Os processos para desenvolvimento do projeto são constituídos a partir da criação de um cluster hospedado na plataforma *Cloud* MongoDB Atlas, os acessos ao ambiente

na nuvem serão todos feitos através da aplicação desktop Mongo *Compass*. Com o uso desta ferramenta algumas simulações serão feitas, como inserção de registros dentro do nó principal do cluster e a manipulação dos nós secundários. Através da utilização do Mongo *Compass* também será realizado o acompanhamento e análise da replicação dos dados. Por fim, será verificada a integridade dos dados replicados e executado testes de alta disponibilidade, ao realizar o desligamento do nó principal e acompanhar a transformação de um dos nós secundários para o status de nó de gerenciamento.

2 Referencial teórico e trabalho correlatos

2.1 Banco de Dados Não Relacional

Embora os bancos de dados relacionais ainda sejam predominantemente usados no mercado de tecnologia da informação, como Oracle Database, Microsoft SQL Server e PostgreSQL. Nos últimos anos a utilização de bancos não relacionais se popularizou, devido aos seus custos-benefícios sobre a escalabilidade e gerenciamento de dados.

Segundo Kaufeld (1996), o modelo de banco de dados relacional possui a capacidade de lidar com grandes volumes de informações, eliminando dados redundantes. Neste modelo, a possibilidade de construção de um relacionamento lógico entre as informações contribui para a redução de registros redundantes armazenados na base de dados, otimizando consultas e processos executados sobre o banco.

A fornecedora líder e independente de soluções de software de integração de dados Informática Corporation (2009) divulgou resultados detalhando os desafios enfrentados pelo mercado da tecnologia da informação, devido ao grande aumento no volume de dados em aplicações. A pesquisa revela a existência de um grande esforço técnico e financeiro para manter o ciclo de vida de uma aplicação que apresenta um crescimento exponencial em sua base. A necessidade de possuir uma infraestrutura composta por servidores robustos e o número elevado de profissionais qualificados para o gerenciamento e manutenção do banco são parâmetros que encarecem consideravelmente o processo de escalabilidade para as estruturas relacionais.

Defronte deste cenário, a pesquisa e desenvolvimento de bancos distribuídos se tornou uma das vias alternativas para evitar os altos custos relacionados a escalabilidade e gerenciamento de bancos relacionais. Segundo Marta Mattoso (2010) a existência de bancos de dados distribuídos, trouxe uma elevação na confiabilidade através de transações distribuídas, pois espera-se que os Sistemas de Banco de Dados Distribuídos (SBDD) ofereçam confiabilidade por trabalharem com componentes replicados eliminando, assim, pontos únicos de falha. Ainda, com os SBDD tem-se aumento de desempenho, facilidade na localização dos dados, o seu compartilhamento de recurso não mais tão crítico levando a redução de tempo pelo acesso remoto aos dados e o seu paralelismo no processamento, além de uma estrutura que contribui para uma escalabilidade menos burocrática.

Segundo Moura e Casanova (1999), a criação de SBDD contribui de forma significativa para o aumento da produtividade em desenvolvimento de aplicações, um fator importante desde longa data. De fato, tais sistemas simplificam a tarefa de definir aplicações que requerem o compartilhamento de informação entre usuários, programas ou organizações onde os usuários da informação, ou mesmo as fontes de informação, estão geograficamente dispersas.

Em 2005, ocorreu o lançamento de um banco de dados não-relacional *open source* chamado CouchDB. De acordo com CouchDB (2020), este banco de dados usa da tecnologia *JavaScript Object Notation* (JSON) para armazenar dados. O JSON é considerado um formato leve para transferência de dados entre computadores. Também se utiliza da linguagem *JavaScript* para consulta com o *MapReduce*, um modelo de programação para processamento de grandes volumes de dados com um algoritmo paralelo e distribuído em cluster. Devido a estes fatores o padrão de banco de dados não relacional começou a ganhar renome entre as grandes empresas de desenvolvimento.

Os bancos de dados não relacionais podem ser categorizados em alguns subtipos. Dentre eles é possível citar os bancos estruturados por Chave/Valor e orientados à documentos ao qual o banco de dados MongoDB implementa e faz parte do contexto deste trabalho.

Segundo a plataforma de serviços de computação em nuvem (AMAZON WEB SERVICES, 2020), um banco de dados de chave-valor armazena dados como um conjunto de pares de chave-valor em que uma chave funciona como um identificador exclusivo. A chave e os valores podem ser qualquer coisa, desde objetos simples

até objetos compostos complexos. Alguns exemplos de bancos que utilizam dessa estrutura são: Azure Table Storage, Redis e Oracle NoSQL. A Figura 1 demonstra a estrutura de armazenamento utilizada por um banco de chave e valor, onde visualizamos o identificador da chave e o valor armazenado na primeira e segunda coluna, respectivamente.

Figura 1 - Exemplo da estrutura de um banco de dados chave/valor

Chave	Valor
livro_4666_ano	2020
livro_4666_pagina	356
livro_4667_ano	1999
livro_4667_pagina	199

Fonte: Elaborado pelos autores, 2020.

Um banco de dados de documentos é um tipo de banco de dados não relacional projetado para armazenar e consultar dados como documentos do tipo JSON. Os bancos de dados de documentos facilitam para que os desenvolvedores armazenem e consultem dados usando o mesmo formato de modelo de documento que usam no código do aplicativo. Alguns exemplos de bancos que utilizam dessa estrutura são: MongoDB, Cassandra e DynamoDB (AMAZON, 2020).

2.2 MongoDB

Segundo Tabet, Mokadem e Laouar (2018) MongoDB é um Sistema de Banco de Dados NoSQL orientado a documento desenvolvido pela 10*gen*, atualmente conhecida como MongoDB. Em 2007 foi escrito na linguagem de programação C++. O sistema gerencia coleções de documentos *Binary* JSON (BSON). O formato de documento BSON é um complemento de documentos do tipo JSON que possui extensões que permitem a representação de tipos de dados que não são parte do formato JSON, possibilitando assim a criação de índices e comparação de objetos com expressões de consulta em outras chaves BSON de nível superior que estão relacionadas.

Já a documentação oficial do fornecedor (MONGODB, 2008a) explica que seu dimensionamento automático afasta a necessidade de gravar scripts ou ter qualquer consultoria para tomar decisões de dimensão. O dimensionamento de seus clusters ajuda a controlar e manter os custos de uma maneira capaz de atribuir um intervalo

entre as camadas dos clusters para que seus clusters possam escalar automaticamente de forma contínua. Resultando em um processo fluido com a redução no tempo de inatividade, o que melhora a performance de sistemas e contribui para sua escalabilidade.

O MongoDB possui tecnologia de fragmentação automática (*sharding*), que distribui os dados para várias máquinas diferentes, fragmentando e separando grandes conjuntos de dados e tarefas, dessa forma, possibilita uma alta performance estrutural (MONGODB, 2008b).

Outra tecnologia utilizada pelo sistema para tolerância a falhas (*failover*) é a criação de uma redundância de dados em uma coleção de réplicas (replica set). Essa coleção de réplicas consiste em um grupo de servidores que mantém o mesmo conjunto de dados, criando uma alta disponibilidade dos dados e redundâncias que possibilitam a rápida recuperação dos dados em casos de falhas (MONGODB, 2021).

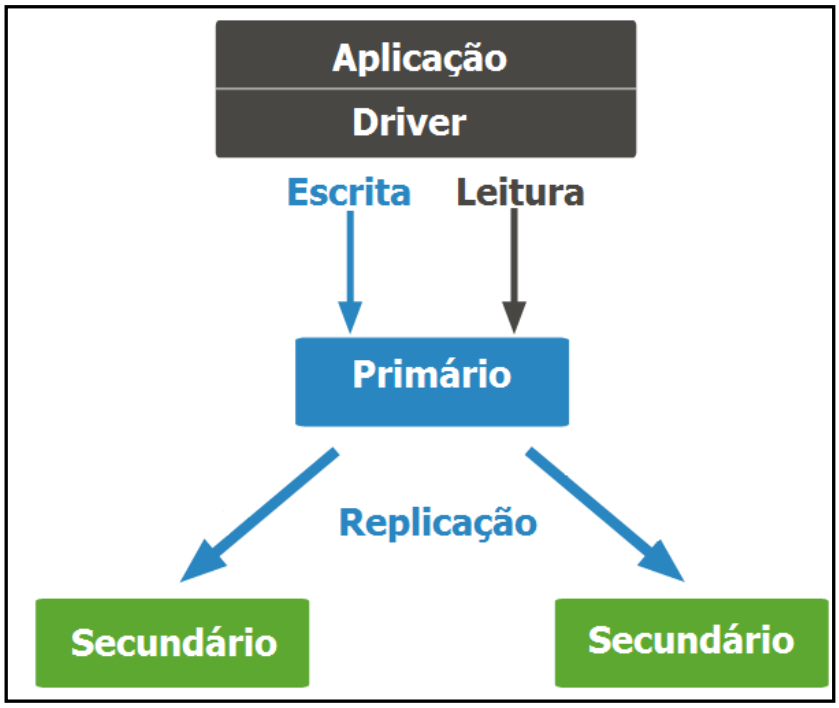
2.2.1 Replica Set

A replicação de dados é constituída a partir de um grupo de processos chamados de *mongod*. Estes processos garantem uma redundância em casos de queda da máquina principal, assim como uma alta disponibilidade das informações.

O processo de replicação é baseado em um grupo de instancias *mongod*. Estas instancias serão utilizadas como nós de armazenamento para a replicação de dados, sendo uma delas categorizada como nó primário e as demais denominadas como secundárias.

Assim, o nó primário é responsável por registrar todos os procedimentos que ocorreram e armazenar essas informações em seu *log* de operações (*oplog*). Já os nós secundários ficam responsáveis pela leitura destes arquivos de *log* e execução das operações ali contidas, assim realizando a replicação (MONGODB, 2016). A Figura 2 demonstra um fluxo simples de como a estrutura de um replica set funciona no banco de dados não relacional MongoDB, onde a aplicação efetua as rotinas de escrita e leitura, sobre o nó primário do cluster, e logo em seguida realiza as rotinas de replicação em seus nós secundários.

Figura 2 - Exemplo do fluxo de um replica set no MongoDB



Fonte: Adaptada de MongoDB Manual, 2020

2.2.2 MongoDB Atlas

A plataforma Atlas é um serviço de nuvem que se responsabiliza pela hospedagem, gerenciamento e a segurança de um cluster MongoDB, instanciando servidores de nuvem em plataformas como: Amazon *Web Services*, Google *Cloud Platform* e Azure. Sendo considerado um serviço de Banco de dados como Serviço (DBaaS), a plataforma tem o intuito de administrar toda a infraestrutura, manutenção e segurança das máquinas e clusters de seus clientes. Todo o gerenciamento da plataforma pode ser feito facilmente através de uma interface *web*, disponibilizada pelo próprio MongoDB Atlas. A plataforma que oferece também um pacote de serviços grátis, chamado de camada M0. Por ser um serviço gratuito, a camada M0 apresenta algumas restrições de recursos como o uso de memória RAM compartilhada, limitação de 512 *megabyte's* de disco e Unidades Centrais de Processamento Virtuais (vCPU's) compartilhadas.

O MongoDB Atlas oferece alguns tipos de clusters que são subdivididos nos seguintes tipos: *Shared Clusters*, que são *clusters* com recursos compartilhados aplicados à camada de recursos grátis da plataforma; *Dedicated Development Clusters*, que são normalmente utilizados para o desenvolvimento de aplicações, porém possuem mais recursos de infraestrutura; e *Dedicated Production Clusters*,

que são máquinas potentes, podendo ter discos e configurações de infraestrutura customizáveis.

2.3 Replicação de Dados

Segundo Coulouris et al. (2013), replicação é a chave para fornecer alta disponibilidade, cuja tendência é direcionada à computação móvel e, consequentemente, a operações desconectadas, e tolerância a falhas em sistemas distribuídos, a qual tem sido uma crescente preocupação para serviços fornecidos em sistemas de segurança críticos e outros tipos de sistemas importantes.

A técnica de replicação permite um ambiente altamente disponível, que de acordo com Ladin, Liskov e Ghemawat (1992), possui dois tipos de operações: operações de atualização que alteram, mas não observam o estado da aplicação; e operações de consulta que observam o estado, mas não o modifica. A replicação distribui a operação de *update*, para que os dados estejam consistentes em todos os bancos e que toda operação de consulta realizada pelos usuários contenha a visualização desses dados atualizados. Além de oferecer uma solução altamente tolerante a falhas, onde o comportamento correto é garantido, independentemente do número e do tipo das falhas. A replicação de dados pode se dividir em alguns tipos.

Segundo MariaDB (2020) a replicação de dados com *binary log* contém o *script* de todas mudanças ao banco de dados, assim como quanto tempo leva para cada declaração ser executada. Ele consiste em um grupo de arquivos *binary* log e também um índice. Isso significa que declarações como *CREATE*, *ALTER*, *INSERT*, *UPDATE* e *DELETE* serão registradas, mas declarações que não tem efeito nos dados, como *SELECT* e *SHOW*, não serão registradas. O propósito do *binary log* é permitir replicação, onde dados são enviados de um ou mais nós principais para um ou mais servidores secundários, baseando-se no conteúdo do *binary log* e assim auxiliar nas operações de *backup*.

De acordo com Dewald (2004), a replicação em *snapshot* captura os dados em um servidor e move esses dados para outro servidor ou banco de dados. Após a *snapshot* inicial de sincronização, a replicação pode atualizar dados em tabelas públicas periodicamente, baseado na programação especificada. Apesar da

replicação em *snapshot* ser o tipo mais fácil de manter e instalar, essa técnica requer que os dados sejam copiados novamente quando a tabela for atualizada.

Dewald (2004) afirma que a replicação transacional envolve copiar dados de um publicador para um inscrito, uma vez que houver transações ocorrentes no publicador, elas são entregues aos inscritos. As cópias iniciais dos dados são transportadas pelo mesmo mecanismo utilizado na replicação em *snapshot*, que realiza a captura de dados em um publicador e o move para seus inscritos. As transações são enviadas aos inscritos de acordo com as inserções, atualizações ou deleções realizadas nos publicadores pelos usuários do banco de dados.

3 Materiais e métodos ou desenvolvimento

Para o desenvolvimento dos experimentos, o Banco de Dados como Serviço (DBaaS) oferecido pela MongoDB, o MongoDB Atlas, será utilizado para a criação de um ambiente clusterizado. Dentro da plataforma, a camada M0 será utilizada, pois é o nível de serviço disponível entregue gratuitamente com algumas restrições. Sua especificação conta com algumas restrições de recursos, com uso de memória RAM e vCPUs compartilhadas e limitação de 512 *Megabytes* de disco. Além disso alguns procedimentos são necessários para a criação do ambiente em nuvem. Foi selecionado o provedor de hospedagem em nuvem, a região e as configurações para o clusters, no caso do projeto, a opção de fornecedor da Google *Cloud*, pois ela oferece hospedagem na região da América do Sul, o que contribuirá para a diminuição da latência das atividades. Para a configuração do ambiente foi necessário a criação de um projeto MongoDB na plataforma Atlas, criação de usuários e o gerenciamento de suas permissões de atuação, criação do cluster computacional e de seu *replica set*, criação de um banco de dados que servirá como base para a funcionalidade de espelhamento e replicação da plataforma. Além da liberação de acesso de rede, para que durante o desenvolvimento seja possível o acesso do ambiente clusterizado através de outras interfaces.

Outros passos também foram necessários. Como a instalação da interface gráfica MongoDB *Compass* que simplifica o trabalho com o MongoDB e suas conexões com o ambiente de desenvolvimento criado na plataforma Atlas. A instalação da ferramenta Robo 3T também foi necessária para trabalhar de forma otimizada com a criação de *collections*, índices e operações de *insert*, *update* e

delete. Trata-se de uma ferramenta gratuita para se trabalhar com o MongoDB de forma gráfica.

As tarefas de execução rotineiras foram feitas a partir de computadores com o sistema operacional *Windows* 10 Pro 64-bit, com processador Intel Core I5, memória RAM de 12 GB e armazenamento de 500 GB. A partir destas máquinas as rotinas de verificação, criação de *collections*, validação das replicações do ambiente clusterizado foram efetuadas.

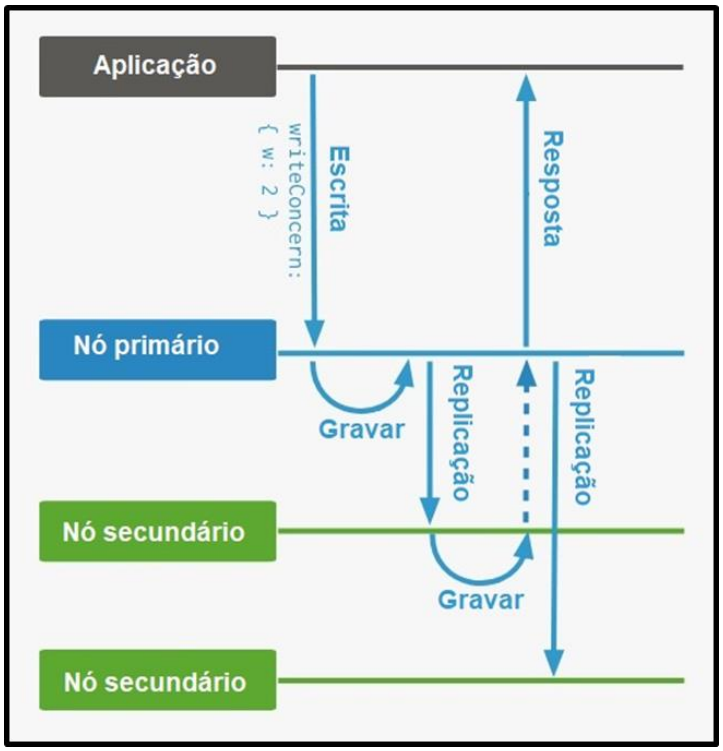
As tarefas realizadas durante o trabalho foram executadas visando a criação de uma estrutura clusterizada a partir de ferramentas e serviços gratuitos e, através desta estrutura, foi possível analisar e testar a rede de clusters criada. Também foi verificada a capacidade de armazenamento dos nós, juntamente com sua performance nas execuções de escrita e leitura de documentos em seu nó principal e durante o processo de replicação. Além de simular um ambiente de falha no nó principal, para analisar a integridade dos dados que foram replicados nas instâncias secundárias.

Para realizar os testes de alta disponibilidade do cluster, um arquivo JSON foi importado no cluster de dados, através da ferramenta Mongo Compass, este arquivo é disponibilizado em uma base de dados pública chamada Portal Brasileiro de Dados Abertos (DADOS.GOV.BR, 2021), no nó primário. As informações escolhidas são referentes a processos de licitações públicas, onde cada registro JSON possui os seguintes atributos: interessado (Solicitante da licitação); sigla do estado onde a solicitação foi realizada; tipo de interessado (Classificando se foi uma solicitação municipal ou estadual); tipo de operação; finalidade; tipo de credor; descrição do credor; valor; número da solicitação; e status. Esta base foi selecionada devido a sua volumetria de 5.57 *Megabytes*, contando com 21.293 documentos JSON e o nível baixo de complexidade dos dados para análise. Como pode-se observar na Figura 5.

O preenchimento do nó primário foi feito através de uma aplicação utilizando o Mongo Compass. Depois desta etapa foi necessário verificar, através da ferramenta gráfica Compass, se os arquivos foram replicados de forma correta nos nós secundários do cluster. Outras análises também foram feitas, como também pode-se observar na Figura 3, uma série de procedimentos são realizados para que a réplica dos dados seja executada, um dos focos do trabalho foi analisar cada uma das etapas ilustradas abaixo. A verificação foi processada durante a execução da

réplica com o intuito de identificar o tempo total necessário para que a replicação se conclua nos nós secundários, verificando a taxa de transferência de dados através das interfaces do MongoDB Atlas. A análise da estabilidade dos nós secundários durante o processo de replicação também foi feita, para levantamento de informações sobre o processo performático da réplica. As análises dos resultados foram realizadas de acordo com a alta disponibilidade encontrada nas aplicações do MongoDB. Após estas etapas o foco foi na análise da alta disponibilidade do cluster, através do painel do Mongo DB Atlas foi executado o desligamento do nó principal e acompanhamento do sistema para verificar qual dos nós secundários será promovido para nó de gerenciamento e se o processo de réplica e armazenamento dos registros não será comprometido. Estas verificações foram feitas através do painel Atlas, consultando as *collections* inseridas dos nós secundários e também através de gráficos que mostram o término da replicação.

Figura 3 - **Fluxo para testes de replicação**



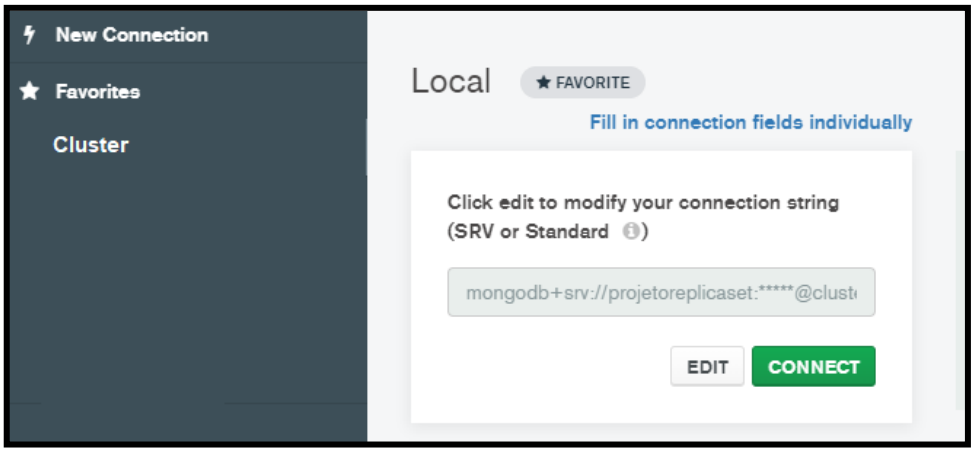
Fonte: Adaptada de MongoDB Atlas, 2020.

4 Resultados e discussão

Foi utilizada uma ferramenta distribuída gratuitamente para realizar a conexão ao cluster, simulando um acesso externo ao banco de dados para realizar os testes

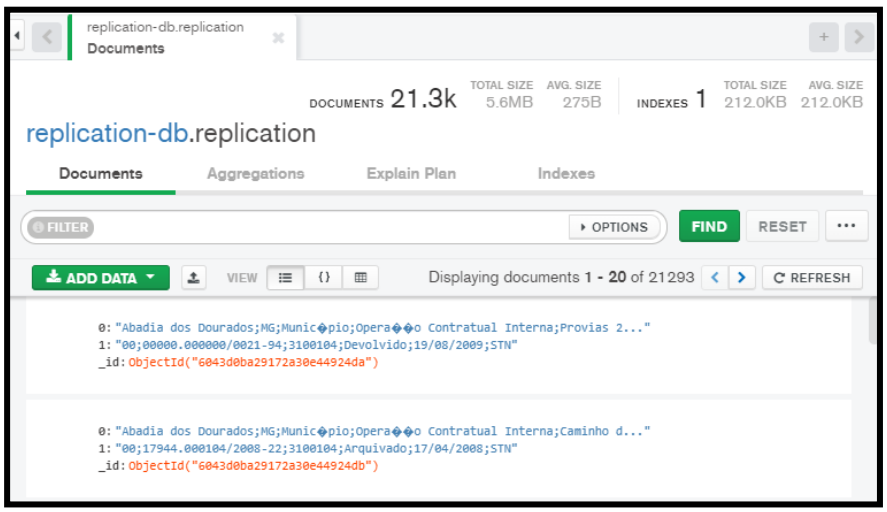
de alta disponibilidade. Para isto, foi instalada uma ferramenta proveniente do MongoDB chamada MongoDB *Compass* em um computador fora do cluster, mas que possui acesso a rede clusterizada em questão. Com o MongoDB *Compass* ativo a conexão foi através ferramenta na aba "*New Connection*" diretamente em seu nó de gerenciamento, como indicado na Figura 4. Com o banco de dados "*replication*-db" e a coleção "*replication*" selecionados, o arquivo JSON foi importado e inserido na estrutura do banco, automaticamente, através da aplicação do MongoDB *Compass* como mostrado na Figura 5.

Figura 4 – Conexão ao cluster



Fonte: Elaborada pelos autores, 2021.

Figura 5 – Importação do arquivo JSON

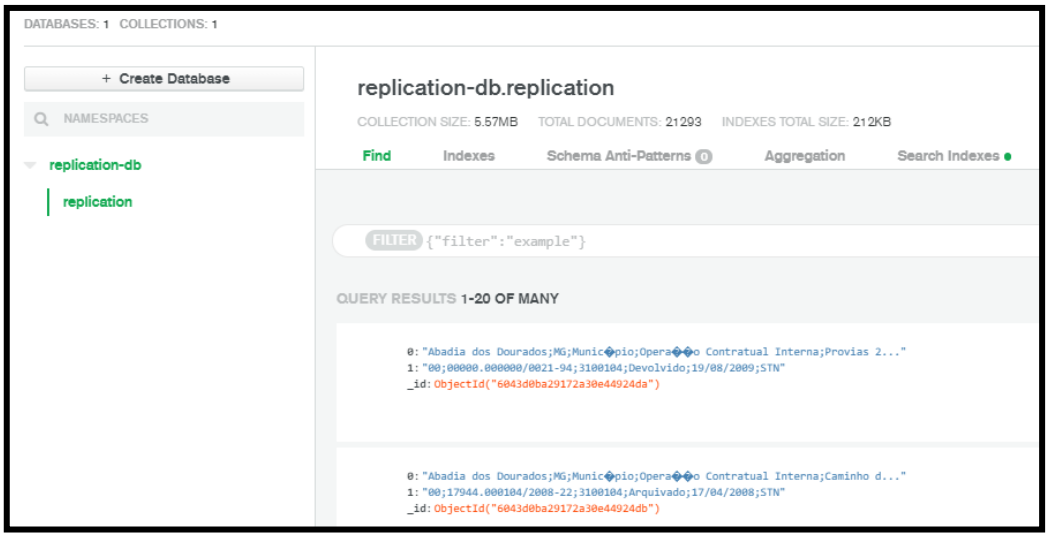


Fonte: Elaborada pelos autores, 2021.

Utilizando o painel da plataforma Atlas é possível acessar a instância do nó de gerenciamento do MongoDB e validar se os registros realmente foram inseridos e

replicados. Neste processo retornamos as informações das coleções com sucesso conforme apresentado na Figura 6. Após a confirmação das inserções na estrutura do nó primário, foi realizada uma configuração sobre os nós secundários através da plataforma Atlas, para que eles adquirissem permissões necessárias para a realização dos testes de replicação, desbloqueando seus status de nós passivos.

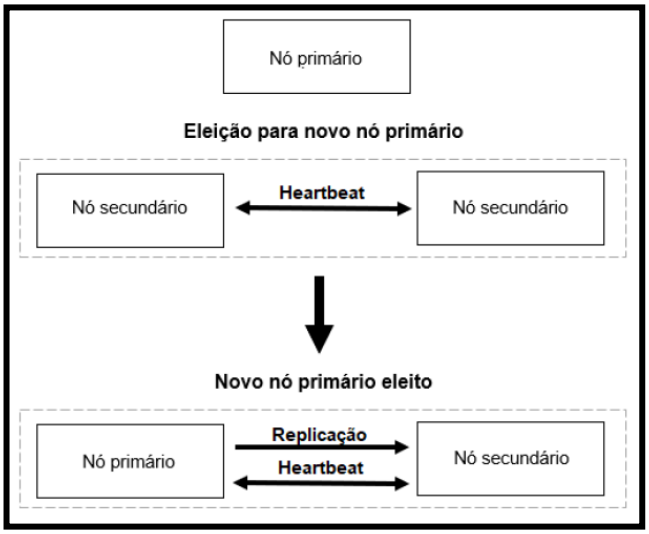
Figura 6 – Dados inseridos e replicados no cluster



Fonte: Elaborada pelos autores, 2021.

Para executar o teste de alta disponibilidade a conexão sobre o nó primário, que interliga os demais nós do cluster, foi interrompida para que um dos nós secundários assumisse o status de nó de gerenciamento. Neste teste foi realizado o desligamento do nó primário que hospedava o nó de gerenciamento o tornando indisponível para a *ReplicaSet* ativa entre os nós secundários restantes. Podemos observar o funcionamento da mudança de nó de gerenciamento na Figura 7.

Figura 7 – Troca de nó de gerenciamento



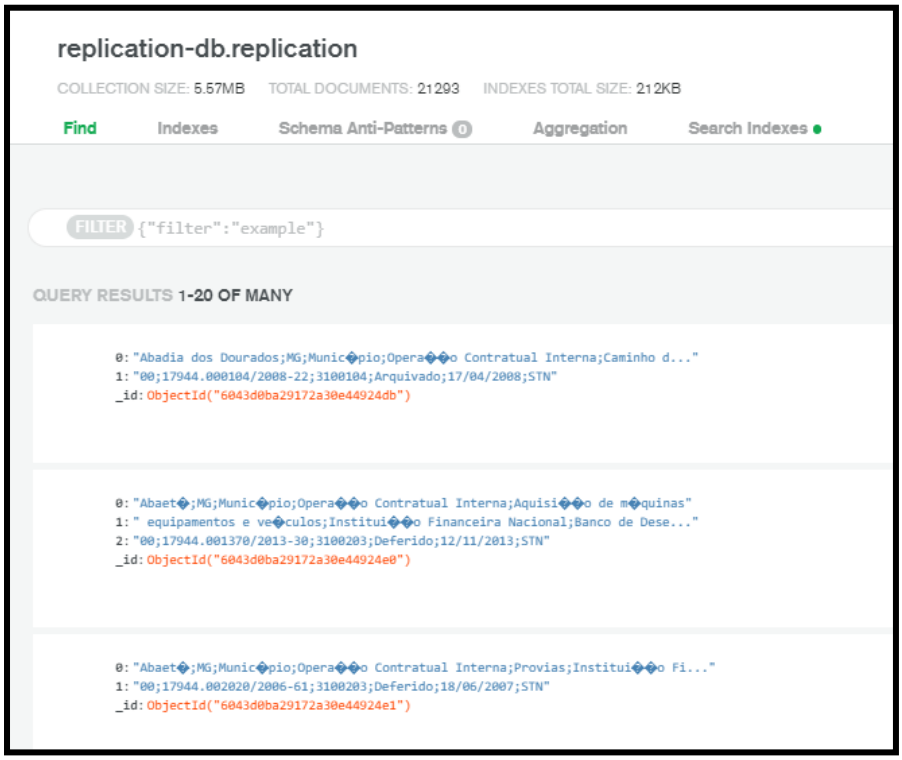
Fonte: Adaptada de MongoDB, 2008d.

A versão 4.0, e superiores, do MongoDB possuem o protocolo de replicação de versão 1, onde a nomeação de árbitros para auxiliar na troca de nó de gerenciamento não é mais necessária. O árbitro do MongoDB possui o dobro de influência na nomeação de um novo nó de gerenciamento, nesta votação cada nó secundário vota em si mesmo com destino para ser promovido a nó de gerenciamento, assim tendo o árbitro a função de desempatar a votação. A partir do MongoDB 4.0 a falta da funcionalidade do árbitro fez com que a detecção e a resolução de indisponibilidade se tornassem mais ágeis. Para verificar qual dos nós foi escolhido para ser promovido a nó primário, podemos acessar a plataforma Atlas e realizar esta verificação através do painel de gerenciamento ao analisar as informações e atualizações sobre as posições atuais dos nós no conjunto de *ReplicaSet*. Durante a execução foi constatado que o nó secundário de número 2 foi promovido para nó de gerenciamento (nó primário), devido a sua informação "*stateStr*" estar nomeada como "*PRIMARY*".

Com a falha na comunicação da máquina em seu antigo nó primário em meio ao conjunto de *ReplicaSet*, ainda conectado na instância do MongoDB, é possível fazer consultas no banco de dados criado, mas não é possível inserir dados neste nó que está indisponível. Para que seja possível testar a integridade do conjunto de *ReplicaSet*, foi iniciada uma nova conexão do MongoDB *Compass* com o atual nó primário (Antigo nó secundário de número 2), sendo executadas outras inserções de

documentos. Após as inserções, foi consultado no nó secundário (Nó de número 3) restante, se os documentos foram replicados em sua base, para isso consultamos as *collections* replicadas utilizando o painel da plataforma Atlas, conforme podemos observar na Figura 8.

Figura 8 – Replicação de dados sobre o nó de número 3

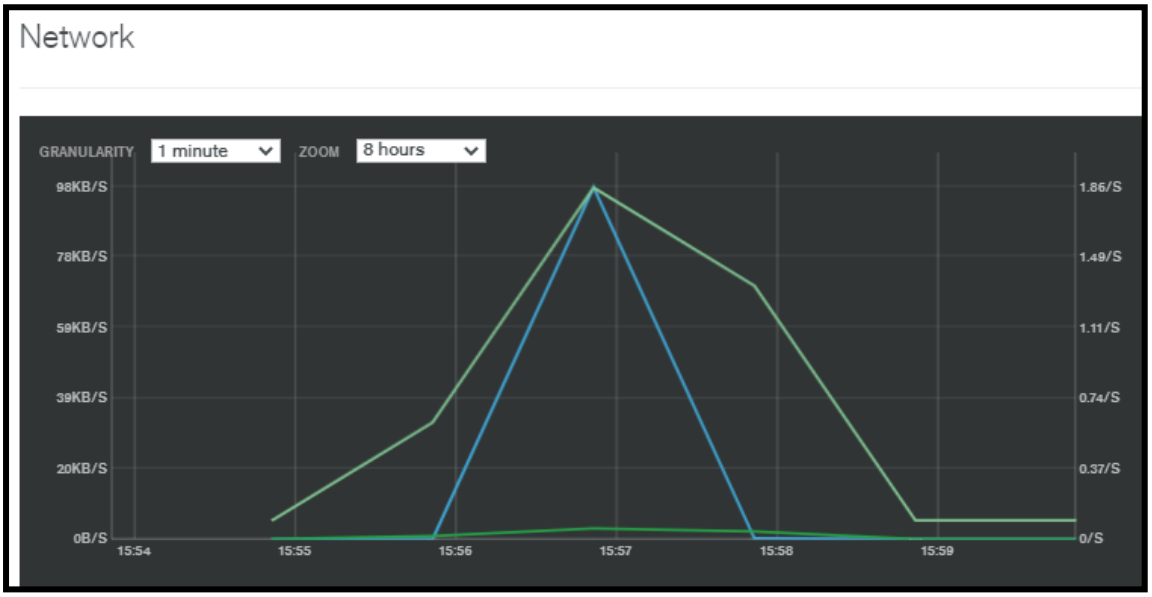


Fonte: Elaborada pelos autores, 2021

Para concluir os testes, foi necessário validar se o nó indisponível (antigo nó de gerenciamento) iria adquirir automaticamente os documentos replicados nos outros nós durante o período em que estava inativo. Depois de sua reinicialização, foi inicializada a instância do MongoDB. Ao passar da fase de inicialização foi realizado as verificações através do painel da plataforma Atlas, executando comandos para obter as informações de posicionamento na hierarquia de *ReplicaSet* e do sucesso do recebimento dos documentos replicados. Por meio da visualização das *collections* do nó foi constatado que as informações que haviam sido acrescentadas nos nós de número 2 e 3 durante a inatividade do nó de número 1 (antigo nó de gerenciamento), foram replicadas no nó de número 1 quando foi reativado. Isto ocorre devido ao recurso de redundância em grupo, que automaticamente faz a replicação de dados dos nós ativos para o nó anteriormente inativo, comparando os itens armazenados no nó primário com o nó recém ligado,

aferindo os *logs* (registros de operações) dos diferentes nós, assim aumentando a disponibilidade dos dados. Referente à análise de conexão com os clusters podemos observar que a transferência de dados através da rede foi realizada em poucos segundos, um resultado de performance satisfatório considerando o fator do uso de uma camada de gerenciamento sem custos. Podemos observar o gráfico da rede na Figura 9.

Figura 9 – Dados referentes ao uso de rede durante a replicação



Fonte: Elaborada pelos autores, 2021

Outra vantagem estabelecida através desta forma de importação e replicação de dados é a utilização da interface gráfica MongoDB *Compass* e dos recursos disponíveis no painel de gerenciamento da plataforma Atlas. Sem estes facilitadores os processos para importação e replicação de dados deveriam ser executados através de linhas de comando, tornando a efetivação da tarefa mais morosa.

Considerações finais

Obtendo os resultados de conclusão foi verificado que o cluster construído e hospedado na plataforma MongoDB Atlas apresentou uma performance satisfatória, analisando o fato do tempo de execução dos processos descritos. Todas as ferramentas e procedimentos destinados ao banco de dados puderam ser implementadas no cluster construído com os microcomputadores. Um cluster

desenvolvido com três dispositivos trouxe as mesmas adequações de gerenciamento em dados, comparado a um construído com outros tipos de hardwares. Micros e pequenas empresas, que não dispuserem de recurso para investimento em máquinas mais robustas, podem optar por uma implementação com micromáquinas. Cada vez mais é necessário possuir uma estrutura que fomente a alta disponibilidade das informações nas empresas. Um ambiente clusterizado traz segurança e estabilidade para o Administrador de Banco de Dados (DBA) de uma empresa, pois os nós apresentam uma estrutura de redundância para controle de perda de dados, caso ocorram falhas em algum dos dispositivos, o que faz com que a empresa confie mais em seu sistema. Um ambiente desenvolvido em uma plataforma criada pelo próprio fornecedor da tecnologia pode ser de grande valia para as empresas que querem melhorar sua estrutura de armazenamento de dados e disponibilidade, trazendo a elas uma forma segura de armazenar suas informações e que pode ser utilizado para implantar indicadores de desempenho na empresa.

Portanto, o cluster construído na plataforma Atlas demonstrou que é possível suportar uma estrutura de banco de dados e mantê-la com uma alta taxa de disponibilidade, replicando seus dados com a utilização de ferramentas como o MongoDB que é disponibilizado por um baixo custo comparadas ao mercado atual. A utilização desta aplicação fica principalmente direcionada para empresas de pequeno porte que estão começando e que querem garantir seu espaço no mercado, podendo realizar escalonamentos horizontais como forma de melhoria ao decorrer do uso e do aumento da demanda.

Com o avanço da tecnologia, tendo em vista possíveis implementações da evolução do projeto, alguns pontos podem ser abordados, como o desenvolvimento de novas estruturas que desencadeiam um aumento no desempenho do projeto. A melhoria das tecnologias de softwares pode influenciar na criação de um projeto similar com mudanças na estrutura das aplicações, como por exemplo, uma proposta de balanceamento das cargas durante a replicação ou até mesmo a criação de um cluster estruturado em uma rede com mais nós.

Referências

AMAZON. Web Services. **Definição de banco de dados de chave-valor**. c2020. Disponível em: https://aws.amazon.com/pt/nosql/key-value/. Acesso em: 7 set. 2020.

AMAZON. Web Services. **O banco de dados de documentos definido**. c2020. Disponível em: https://aws.amazon.com/pt/nosql/document/. Acesso em: 7 set. 2020.

CANALTECH.COM. **Indisponibilidade no e-commerce pode gerar prejuízo de até US$ 8 mil por minuto.** Publicado em: 13 de dez. de 2013. Disponível em: https://canaltech.com.br/e-commerce/Indisponibilidade-no-e-commerce-pode-gerar-prejuizo-de-ate-US-8-mil-por-minuto. Acesso em: 2 out. 2020.

CASANOVA, Marco; MOURA, Arnaldo. **Princípios de Sistemas de Gerência de Bancos de Dados Distribuídos: Edição Revisada.** 1999. Disponível em:http://www.inf.puc-rio.br/~casanova/Publications/Books/1985-BDD.pdf. Acesso em: 7 set. 2020.

COUCHDB. CouchDB. **Guia do usuário**. c2020. Disponível em: https://docs.couchdb.org/en/stable/intro/index.html Acesso em: 2 out. 2020.

COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim; BLAIR Gordon *et al*. **Distributed Systems: Concepts and Design**. 2º. ed. rev. 1988: Addison-Wesley, 1988. p.766

DB-ENGINES. **Knowledge base of relational and NoSQL Database Management Systems.** Disponível em: https://db-engines.com/. Acesso em: 2 out. 2020.

DEWALD, Baya. **Introduction to Database Replication.** Disponível em: https://www.informit.com/articles/article.aspx?p=169612&seqNum=2. Acesso em: 7 set. 2020.

G1.COM. **Fora do ar, Amazon sofre prejuízo estimado em US$ 31 mil por minuto.** 2008. Disponível em: http://g1.globo.com/Noticias/Tecnologia/0,,MUL592388-6174,00FORA+DO+AR+AMAZON+SOFRE+PREJUIZO+ESTIMADO+EM+US+MIL+POR+MINUTO.html Acesso em: 2 out. 2020.

INFORMATICA CORPORATION**. Information, Unplugged: 2009 OAUG Research Line Survey on Enterprise Application Information Lifecycle Management.** ago. 2020: Disponível em: https://vox.veritas.com/legacyfs/online/veritasdata/01947_app-info-lifecycle-mgmt_br_en-US.pdf. Acesso em 09 set. 2020.

KAUFELD, **J. Access 95 para Windows para leigos: Um manual para novos usuários**. LUDEMIR, J. São Paulo: Berkeley Brasil, 1996. 352 p.

LADIN, Rivka; LISKOV, Barbara; GHEMAWAT, Sanjay *et al.* **Providing High Availability Using Lazy Replication**. 10º. ACM TransactIons on Computer Systems, 1992. p.362. Disponível em: https://www.cs.princeton.edu/courses/archive/fall19/cos418/papers/lazy.pdf. Acesso em: 06 set. 2020.

MARTA MATTOSO. **Especialização em Engenharia de Software**, 2010. p.12 - 14 Disponível em: https://www.cos.ufrj.br/~marta/BdDistribuido.pdf. Acessado em 2 de out. 2020.

MARIADB. **Overview of the Binary Log**. 01 ago. 2019: Disponível em: https://mariadb.com/kb/en/library/overview-of-the-binary-log. Acesso em 07 set. 2020.

MONGODB. MongoDB. **Sharding**. c2008b. Disponível em: https://docs.mongodb.com/manual /sharding/. Acesso em: 19 set. 2020.

MONGODB. MongoDB. **Get Started with Atlas**. c2008a. Disponível em: https://docs.atlas.mongodb.com/getting-started/. Acesso em: 19 set. 2020.

MONGODB. MongoDB. **Replica Set High Availability.** C2021. Disponível em: https://docs.mongodb.com/manual/core/replica-set-high-availability/. Acesso em: 11 mar. 2021

NEWS.COMSCHOOL. **Impacto de um Aplicativo Fora do Ar**. 2015. Disponível em: https://news.comschool.com.br/impacto-de-um-servico-fora-do-ar/. Acesso em: 2 de out. 2020

PITANGA, Marcos. **Construindo Supercomputadores com Linux**. 3º. ed. Rio de Janeiro BR: Brasport, 2003. p.25 – 26

REIS, Adriele; SOARES, Claudio; FERREIRA, Jorge; ALMEIDA, Júlio; SILVA, Matheus; GAVINIER, Sabrina. **Cluster de Alta Disponibilidade.** Faculdade de Tecnologia de Guaratinguetá (FATEC-GT) Guaratinguetá, 2011.

SADALAGE, Pramod; FOWLER, Martin. **NoSQL distilled: a brief guide to the emerging world of polyglot persistence.** Upper Saddle River, USA: Pearson Education, 2013.

STALLINGS, William. **Arquitetura e Organização de Computadores**. 8º. ed. São Paulo, BR: Pearson Pratice Hall, 2010. p.532

TABET, Khaoula; MOKADEM, Riad; LAOUAR, Mohammed; **Towards a New Data Replication Strategy in MongoDB Systems.** 2018. Disponível em: https://www.researchgate.net/publication/329847877_Towards_a_New_Data_Replication_Strategy_in_MongoDB_Systems/. Acesso em: 19 set. 2020

TANEMBAUM, Andrew; VAN STEEN, Maarten**. "Distributed Systems: principle's and paradigms**". 2°. ed. 2006.