South American Development Society Journal

Vol.: 04, | Nº.: 11 | **Ano:** 2018 | **ISSN:** 2446-5763 | **DOI:** 10.24325/issn.2446-5763.v4i11p298-320
Data de Submissão: 25/06/2018 | **Data de Publicação:** 24/08/2018

BANCO DE DADOS NO-SQL X BANCO DE DADOS SQL: ESTUDO DE DESEMPENHO EM GRANDES MASSAS

NO-SQL DATABASE X SQL DATABASE: PERFORMANCE STUDY IN LARGE MASSES

Moacir de Souza Oliveira - **Faculdade de Tecnologia da Zona Leste**

Nicole de Farias Melo - **Faculdade de Tecnologia da Zona Leste**

Leandro Colevati dos Santos - **Faculdade de Tecnologia da Zona Leste, Universidade de São Paulo**

Wellington Pinto de Oliveira - **Faculdade de Tecnologia da Zona Leste**

Resumo

Estamos na era das grandes massas de dados. O armazenamento e a manipulação de grandes quantidades de dados são um desafio. Houve uma época em que os bancos de dados do modelo relacional eram a única solução viável, por serem seguros e de fácil manipulação. Com os avanços tecnológicos e a popularização da internet em todo o mundo, a cada instante um número considerável de novos dados são gerados, devido a esse fenômeno surgiram novas abordagens de manipulação de dados, a fim de atender às crescentes necessidades do mercado. Os bancos de dados NoSQL estão sendo cada vez mais reconhecidos como alternativas ao modelo relacional para manipulação de dados. Ambas as abordagens são boas para determinadas situações. Este trabalho tem como objetivo fazer um estudo comparativo entre banco de dados relacionais e banco de dados NoSQL. A nossa pesquisa compara os dois bancos de dados para ajudar o desenvolvedor a identificar qual plataforma é mais adequada para uma massa de dados específica e, assim, auxiliar na tomada de decisão com relação a qual a melhor escolha para o projeto em

questão. A metodologia usada nesse trabalho é a análise das arquiteturas dos bancos de dados e teste envolvendo grandes massas de dados para a verificação do desempenho de cada banco. Nos resultados obtidos na nossa pesquisa observamos uma diferença surpreendente entre o banco de dados MS SQL Server e o MongoDB no tempo de inclusão de dados. Outro resultado que nos chamou a atenção fora o tempo de consulta entre o MongoDB e o MSSQL, que se mostrou significativo para nossas conclusões sobre que banco é mais recomendável para grandes requisições. Também notamos que há diferença no consumo de recursos de cada SGBD, podendo influenciar na escolha de uma das plataformas.

Palavras-Chave: *Computation Offloading*, Processamento, Dispositivos Móveis, Android

Abstract

We are in the era of large bodies of data. Storage and handling large amounts of data is a challenge. There was a time when the databases of the relational model were the only viable solution, by them being safe and easy handling. With technological advances and the popularization of the internet around the world, every moment a considerable number of new data is generated, due to this phenomenon some new data manipulation approaches arise to meet the increasing needs of the market. The NoSQL databases are being increasingly recognized as alternatives to relational model for data manipulation. Both approaches are good for certain situations. This work aims to make a comparative study between relational databases and NoSQL database. Our research compares the two databases to help the developer to identify which platform is best suited for a specific data mass and thus assist in decision-making with respect to what is the best choice for the project in question. The methodology used in this work is the analysis of database architectures and testing involving large masses of data to check the performance of each database. In the results obtained in our research we observed a surprising difference between the MS SQL Server database and the MongoDB in the data inclusion time. Another result that caught our attention was the query time between MongoDB and MSSQL, which proved to be significant for our conclusions about which bank is most recommendable for large requests. We also

noticed that there is a difference in the resource consumption of each DBMS, which may influence the choice of one platform.

Introdução

Atualmente até mesmo tarefas simples estão sendo automatizadas e o número de sistemas com grandes volumes de dados não para de crescer.

No mundo corporativo informação é poder e, pensando nisso, cada vez mais tem sido investido em hardwares e softwares dedicados ao armazenamento e, temos uma ampla gama de SGBDs (Sistema de Gerenciamento de Banco de Dados), desde soluções tradicionais, como bancos de dados SQL a soluções alternativas como os bancos NoSQL.

No meio de tantas possibilidades como saber qual utilizar? Esse trabalho analisa as arquiteturas de banco de dados relacional e de banco de dados NoSQL. O conceito de que os bancos relacionais são adequados a todas as situações ainda é fortemente difundido no mundo corporativo e educacional, devido a características como facilidade de entender e programar, atender bem as necessidades, ter uma estrutura com representações próximas da realidade, ser uma linguagem consolidada, dentre outras.

Muitos não sabem ao certo em quais situações é vantajoso aplicar as alternativas ao modelo relacional. Há poucas recomendações indicando em que contexto usar determinado paradigma, devido a isso surgiu a seguinte questão: Em qual situação usar o paradigma de armazenamento SQL ou o NoSQL?

Esse trabalho tem por objetivo analisar e testar os bancos de dados relacional e NoSQL no quesito desempenho, esperamos que o resultado desse estudo comparativo seja de ajuda aos desenvolvedores que precisam decidir que tipo de banco será apropriado para o projeto que vão desenvolver ou para aqueles que já possui uma aplicação mas desejam migrar para outro banco que atenda melhor as necessidades de alta demanda de escalabilidade do seu sistema.

Fundamentação Teórica

Nesta seção apresentaremos o embasamento teórico para o entendimento das teorias envolvidas neste artigo, definindo, assim, Banco de Dados Relacional e Banco

de Dados NoSQL, seus componentes e WebService e suas apresentações SOAP e REST

Banco de Dados Relacional

Um banco de dados é uma ferramenta para armazenar e organizar informações. Banco de Dados Relacional é todo Banco de Dados cujo armazenamento é realizado em relações de tabelas. Cada relação é composta de tuplas (ou registros) e atributos (ou colunas). Cada registro na tabela é identificado por um campo chave que contém um valor único.

"A relação entre duas tabelas é estabelecida através de valores correspondentes de um campo compartilhado".**Erro! Fonte de referência não encontrada.** As tabelas relacionam-se umas às outras através de chaves. Uma chave é um conjunto de um ou mais atributos que tornam um registro único.

Temos dois tipos de chaves:

1) Chave primária: (PK - Primary Key) é a chave que identifica cada registro. A chave primária nunca se repetirá.
2) Chave Estrangeira: (FK - Foreign Key) é a chave formada através de um relacionamento com a chave primária de outra tabela. Define um relacionamento entre as tabelas e pode ocorrer repetidas vezes. Caso a chave primária seja composta na origem, a chave estrangeira também o será.

Um sistema gerenciador de banco de dados (SGBD) é um sistema de software que define a maneira como os dados serão armazenados e como serão manipulados. " O que difere os SGBD's é o conjunto de requisitos e funcionalidades que eles oferecem, como segurança, integridade, controle de concorrência e recuperação/tolerância a falhas."**Erro! Fonte de referência não encontrada.**.

Linguagem SQL

SQL é a linguagem padrão para se trabalhar com bancos de dados relacionais. Com essa linguagem é possível interagir com o SGBD e realizar as operações necessárias. A linguagem SQL é dividida em 4 agrupamentos.**Erro! Fonte de referência não encontrada.**

DDL - Linguagem de Definição de Dados

Nesse subconjunto encontramos os comandos usados para se definir as estruturas dos dados, a partir desses comandos é possível criar índices, linhas, colunas e tabelas. Os comandos DDL são:

a) CREATE – Esse comando é útil para criar uma nova tabela, uma visão de uma tabela, ou outro objeto;

b) ALTER – Esse comando é responsável por alterar qualquer objeto que foi criado;

c) DROP – Esse comando pode apagar um objeto do banco de dados.

DML - Linguagem de Manipulação de Dados

Desse subconjunto fazem parte os comandos utilizados para se recuperar, incluir, remover, pesquisar ou modificar alguma informação no Banco de Dados:

a) INSERT – Com esse comando podemos adicionar dados em uma tabela;

b) UPDATE – Com esse comando é possível atualizar os dados;

c) DELETE – Com esse comando apagamos dados;

d) SELECT – Considerado o comando mais importante, esse comando é responsável por realizar as consultas.

DCL - Linguagem de Controle de Dados

Esse subconjunto é formado pelos comandos relacionados com a segurança interna do banco de dados, através desses comandos o administrador de banco de dados pode controlar os aspectos de autorização de dados e licenças de usuários, ou seja, decidir quem terá acesso aos dados e quem poderá manipula-los. DCL possui dois comandos:

1) GRANT: Concede permissões a um ou mais usuários e determina as regras para tarefas determinadas;

2) REVOKE: revoga permissões dadas por um GRANT.

DTL - Linguagem de controle de transações

Esse subconjunto possui três comandos para o controle de transações no banco de dados. São eles:

a) BEGIN TRANSACTION - para iniciar a transação;

b) COMMIT - que efetiva as alterações;

c) ROLLBACK - que cancela as alterações.

Bancos de Dados NoSQL

Com o avanço da tecnologia houve um aumento considerável de pessoas e dispositivos conectados, o que fez com que o trabalho de armazenamento de dados fosse revisado otimizando escalabilidade e custos de manutenção desses dados.

Bancos de dados relacionais escalam, mas quanto maior o tamanho, mais custoso se torna essa escalabilidade, seja pelo custo de novas máquinas, seja pelo aumento de especialistas nos bancos de dados utilizados. Já os não relacionais, permitem uma escalabilidade mais barata e menos trabalhosa, pois não exigem máquinas extremamente poderosas e sua facilidade de manutenção permite que um número menor de profissionais seja necessário. **Erro! Fonte de referência não encontrada.**

Os bancos de dados não relacionais passaram a ser conhecidos como NoSQL (sigla para "Not only SQL”). Essa tecnologia não tem como objetivo substituir os bancos de dados relacionais, mas apenas propor algumas soluções que em determinados cenários são mais adequadas. Desta forma, é possível trabalhar com tecnologias NoSQL e banco de dados relacionais dentro de uma mesma aplicação. **Erro! Fonte de referência não encontrada.**

O NoSQL surgiu da necessidade de uma performance superior e de uma alta escalabilidade e esse tipo de banco de dados é uma boa alternativa para lidar com grandes volumes de dados, armazenar estruturas dinâmicas ou lidar com estruturas não convencionais como grafos. Eles também são muito tolerantes a erros.

Os bancos de dados NoSQL são caracterizados por afastar a complexidade dos bancos SQL. A lógica de validação, controle de acesso, mapeamento de consultas de dados indexados, correlação entre os dados relacionados, resolução de conflitos, manutenção de restrições de integridade (constraint), e procedimentos engatilhados são movidos para fora da camada de banco de dados. Isto permite o foco em performance e escalabilidade. Uma das grandes vantagens da arquitetura NoSQL é que a lógica pode ser codificada em qualquer linguagem de programação, ao invés de depender da grande variedade de APIs complexas em um servidor SQL.

Os bancos NoSQL são subdivididos pelo seu núcleo, ou seja, como ele trabalha com os dados. Alguns dos principais núcleos são: Armazenamento em Colunas Amplas/Famílias de Colunas; Armazenamento Orientado a Documentos; Armazenamento Chave-Valor/Tuplas; Eventually Consistent Key Value Store; Banco de Dados Orientado a Grafos; Banco de Dados Orientado a Objetos; Grid & Cloud Database; Banco de Dados XML.

Os núcleos mais utilizados atualmente são Armazenamento Orientado a Documentos; Armazenamento Chave-Valor.**Erro! Fonte de referência não encontrada.**

Banco de dados que trabalham no esquema chave/valor (Key/Value)

Os bancos de dados que trabalham no esquema chave/valor armazenam objetos indexados por chaves, e possibilitam a busca por esses objetos a partir de suas chaves. Esses tipos de bancos de dados são os que aguentam mais carga de dados e tem a maior escalabilidade.

Alguns bancos que utilizam esse padrão são: DynamoDb, Couchbase, Riak, Azure Table Storage, Redis, Tokyo Cabinet, Berkeley DB, etc. **Erro! Fonte de referência não encontrada.**

Bancos de dados orientados a documentos

Baseado em documentos XML ou JSON, podem ser localizados pelo seu id único ou por qualquer registro que tenha no documento. Coleções de atributos e valores, onde um atributo pode ser multi-valorado.

Em geral, os bancos de dados orientados a documento não possuem esquema, ou seja, os documentos armazenados não precisam possuir estrutura em comum.

Essa característica faz deles boas opções para o armazenamento de dados semiestruturados. Alguns bancos que utilizam esse padrão são: MongoDb, CouchDB, RavenDb, etc.**Erro! Fonte de referência não encontrada.**

JSON

JSON é uma estrutura de dados universal, ele é composto de duas estruturas:

1) Um conjunto de chaves/valores. ({ Rua: “Águia de Haia” }, { Modelo: “Fiesta Hatch” });

2) Uma lista ou vetor ordenado. [true, { Rua: “Águia de Haia”}, { Modelo: “Fiesta Hatch” })] ;

“JSON é um formato mais leve para tráfego de dados, sua origem vem do JavaScript, tanto é que JSON significa: JavaScript Object Notation.” “JSON pode ser usado em grande partes das linguagens de programação como Java, PHP, Delphi, Lua, Python e etc. Suas API’s prontas são bem mais simples de utilizar comparada as API’s de XML”.**Erro! Fonte de referência não encontrada.**

JSON é amplamente suportado em praticamente todas as linguagens modernas, graças a sua simplicidade e facilidade de leitura tanto humana quanto computacional. Outro ponto a destacar no JSON é mesmo permitindo flexibilidade em sua estrutura, a velocidade na execução e transporte de dados resultam em duração idêntica a outras formas de armazenamento estruturadas como arquivos XML.**Erro! Fonte de referência não encontrada.**

Web Service

Web Service é a tecnologia que permite às aplicações enviar e receber mensagens independente de sistema operacional e de linguagem de programação. As aplicações podem ser escritas em linguagens diferentes, pois serão traduzidas para uma linguagem universal, o formato XML. **Erro! Fonte de referência não encontrada.**

O principal protocolo de transporte de mensagens dos Web Services é o HTTP, o que ajuda a evitar problemas com firewalls, sendo apenas uma das características positivas dessa tecnologia. **Erro! Fonte de referência não encontrada.**

SOAP (Simple Object Access Protocol)

SOAP é um protocolo de transferência de mensagens em formato XML. O SOAP age como um tipo de framework que permite a intercomunicação entre diversas plataformas através de mensagens.

Aplicando este padrão em Web Services, geralmente o WSDL é usado para descrever a estrutura das mensagens SOAP e as suas ações. Esse padrão serve para que, a partir de um objeto, seja possível serializar o mesmo para XML e assim enviar a mensagem, também, deserializá-lo de volta para o formato original para interpretar a mensagem. **Erro! Fonte de referência não encontrada.**

REST (Representational State Transfer)

REST é outro protocolo de comunicação, mas, diferente do SOAP que possui protocolos bem definidos e um conjunto de regras bem estabelecidas, o REST é considerado apenas uma abordagem arquitetural, ou seja, as mensagens não têm que seguir um formato padrão, há restrições apenas para o comportamento dos componentes envolvidos.

A maior vantagem do REST é sua flexibilidade, o desenvolvedor pode escolher qual formato é o mais adequado para as mensagens do sistema de acordo com sua necessidade, podendo usar formatos como JSON, XML e texto puro, ou outro formato desejado.

Geralmente web services REST são mais leves e, portanto, mais rápidos. Sua desvantagem surge por causa de suas vantagens. A definição das estruturas de dados fica totalmente a cargo do desenvolvedor, assim, problemas de interoperabilidade são frequentes.**Erro! Fonte de referência não encontrada.**

Materiais e Métodos

Esta seção descreve a metodologia e os materiais utilizados para desenvolver esse estudo.

Materiais

As ferramentas utilizadas para o desenvolvimento desse estudo foram o Microsoft SQL SERVER, MongoDB e Node.JS

Microsoft SQL SERVER®

Esse sistema gerenciador de banco de dados relacional usa a Linguagem SQL para definição, acesso e manipulação de dados. O SQL Server é um produto da Microsoft completo e um dos principais bancos de dados do mercado.

Assim como os demais bancos relacionais, o SQL Server usa os comandos básicos da linguagem SQL e também possui uma versão estendida dessa linguagem, conhecida como T-SQL (Transact-SQL). Essa versão personalizada da linguagem SQL oferece “um suporte exclusivo de funcionalidade à sua plataforma”**Erro! Fonte de referência não encontrada.**. Com o SQL Server é possível conectar-se a outros bancos de dados, importar e exportar dados, gerar relatórios profissionais dinâmicos, criar tarefas automatizadas que são acionadas quando as condições especificadas acontecem, dentre outras coisas.

MongoDB

O MongoDB é um SGBD distribuído, orientado a documento, livre de esquemas de armazenamentos, como CouchBase e CouchDB. A estrutura dos documentos possui o formato JSON, um arquivo do tipo javascript, permitindo estruturar itens com nomes e seus respectivos valores e também estruturar arrays, objetos e documentos embutidos. Essa estrutura ocupa pouquíssimo espaço e facilita a leitura do desenvolvedor ao manipular documentos, ao mesmo tempo permite a portabilidade dos dados para outras estruturas de armazenamento, internamente os dados são armazenados em Binários - JSON (BSON).

O projeto MongoDB, escrito em C++, teve início em 2007, mas o Banco de Dados somente foi concluído em 2009 lançando assim a primeira versão do MongoDB nesse ano. "Diversas linguagens e plataforma já possuem drivers para o MongoDB, entre elas destacam-se: C, C#, C++, Haskell, Java, JavaScript, Perl, PHP, Python, Ruby e Scala."**Erro! Fonte de referência não encontrada.**

Há algumas coisas que merecem consideração, dentre elas a capacidade do journaling de realizar logs (registros) antes de qualquer modificação, caso ocorra algum erro durante uma modificação, esse registro contém todas as interações dos dados e com esse rastreio esses dados seriam restaurados **Erro! Fonte de referência não encontrada.**. O framework MapReduce auxilia o MongoDB nas agregações de

grandes volumes de dados, ele leva o nome de ‘Mapper’ por criar um mapa dos dados e o nome de ‘Reducer’ por reduzir os dados gerados de forma mais simples. O MapReduce também permite processar dados e agregações usando querys com a cláusula GROUP BY. **Erro! Fonte de referência não encontrada.**

Sharding é a abordagem do MongoDB para atender as demandas de crescimento de dados, nela os registros de dados são armazenados em várias máquinas e isso permite a aceleração na gravação dos dados através de replicação e baixa latência nas consultas, fornecendo alta disponibilidade dos dados. Com autosharding o MongoDB proporciona maior proteção aos dados nas replicações, a arquitetura do autosharding é dividida em três partes: roteadores de consulta, conjuntos de sharding e os servidores de configuração **Erro! Fonte de referência não encontrada.**. Outras características:

a) Querys Ad hoc: consulta de uso especifico com retorno definido, ideal para determinados momentos em que se espera dados representando a instância dos dados;

b) Indice: Qualquer campo de um documento, array ou documento integrado, pode ser indexado aumentando a performance da consulta. MongoDB também suporta índices secundários.

c) Balanceamento de Carga: O escalamento horizontal é feito com

sharding, o que permite a distribuição do mesmo dado entre os servidores permitindo a leitura dos dados duplicados nos fragmentos escravos.

d) JavaScript: O mongoDB permite a execução de códigos .js diretamente no servidor e é sua linguagem nativa de manipulação.

NodeJS

O Node é um programa de servidor com um modelo inovador. Com o Node é possível atender a requisições de dezenas de milhares de conexões simultaneamente, devido a sua arquitetura que não precisa de um encadeamento e alocar memória para cada conexão. Com o Node “ cada conexão cria um processo, que não requer que o bloco de memória o acompanhe.” **Erro! Fonte de referência não encontrada.**

O Node usa o mecanismo V8 JavaScript e o redireciona para uso no servidor. JavaScript sendo executado no servidor é uma ideia recente, visto que a natureza do JS é a execução no lado do cliente. Seu funcionamento ocorre com a interpretação do código fonte e posteriormente a execução, o V8 é escrito em C++ e com esse mecanismo foi possível alcançar desempenho ultrarrápido, e está disponível para baixar e integrar ao sistema desejado. **Erro! Fonte de referência não encontrada.**

Métodos

Após definir o que fora utilizado para o projeto, é importante definir quais foram os dados utilizados para alimentá-lo, para que assim ele possa funcionar, portanto, nesse capítulo serão abordados os dados que entraram nas aplicações servidora e cliente, a execução de parse (analise textual) e os resultados referente a desempenho em cada banco de dados.

Massa de Dados

Para esse projeto, fora utilizado massa de dados climáticos dos sites ClimaTempo e TempoAgora referente ao período presente, previsão mais recente para a hora corrente, de aproximadamente dezoito dias, a cada hora somente é possível extrair 6335 registros ou documentos, visto que no ClimaTempo e TempoAgora estão disponíveis 977 e 5358 municípios respectivamente.

Essa massa de dados foi obtida através do download do arquivo HTML de cada página climatológica e a partir disso foi extraído os dados dos cabeçalhos HTML. Ao todo é possível obter 152.040 previsões no período de 24 horas. Foram usadas duas máquinas cliente com as configurações de hardware conforme os quadros 4 e 5. Para o link de conexão com a internet fora usado um link de fibra ótica de 35 Mbps e outro link de 4 Mbps de ADSL, ambos prestados pela VIVO.

Os quadros 1 e 2 apresentam as configurações dos clientes

Quadro 1. Configuração do Cliente I

Hardware/Software	Descrição
Processador	Intel Core i3 CPU M 550 @ 3.20GHz
Memória RAM	4GB RAM DDR3 1,333 GHz
Espaço em HD	ST32000542AS
Sistema Operacional (64 bits)	Ubuntu 14.04

Quadro 2. Configuração do Cliente II

Hardware/Software	Descrição
Processador	Intel Xeon CPU E5310 @ 1.60GHz
Memória RAM	8GB RAM DDR2 667 Mhz
Espaço em HD	TOSHIBA MK5065GS
Sistema Operacional (64 bits)	Ubuntu 16.04 LTS

Aplicação Cliente – Servidor

Para esse estudo usamos duas aplicações, uma aplicação cliente e uma aplicação servidora em conjunto com os bancos de dados MongoDB e MS SQL Server.

Aplicação Cliente

A aplicação cliente possui a função de requisitar dados de municípios nos web services ClimaTempo e TempoAgora, a fim de processar os dados climáticos. Deve - se destacar que os web services ClimaTempo e TempoAgora não possuem todas as funções de web services, seja RestFull ou SOAP. Mas, ambos atuam como webs services para nosso sistema por fornecer acesso aos dados através de requisições HTTP retornando assim uma página web HTML e, a partir disso a aplicação cliente trata esse documento HTML e extrai as informações relevantes ao clima. Com essas informações a aplicação cliente se comunica com a aplicação servidor através de uma interface de comunicação com o Node,js.

Aplicação Servidora

A aplicação servidora atua como mediadora entre os bancos de dados e a aplicação cliente, dessa forma ela atua de forma transparente tanto por requisições para o banco de dados MS SQL Server como também para o MongoDB e, através dela as requisições feitas pela aplicação cliente são processadas, algumas dessas funções são: verificar municípios disponíveis para a inclusão de novas previsões, alteração de cidades disponíveis para o estado verificado e armazenamento de previsão.

Ao responder as requisições é processado internamente funções parse (análise de texto), geração de dados referentes a data e o horário padronizado para ambos os bancos de dados, já ao processar alguma operação relacionada a banco de dados é analisado o tempo de execução da mesma. As configurações de hardware e sistema operacional é descrita no quadro 3.

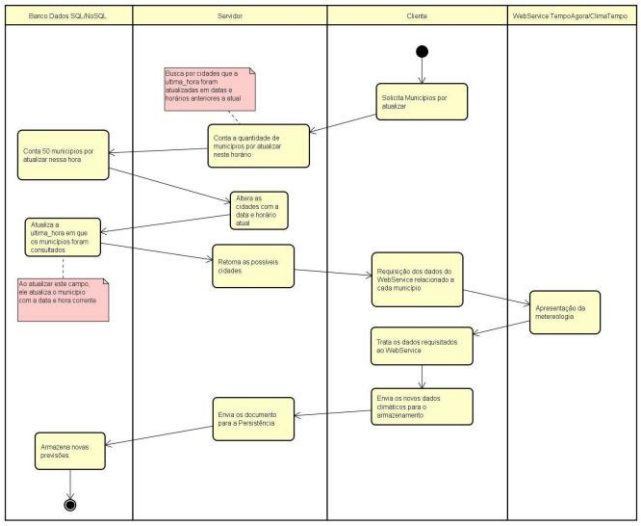
Quadro 3. Configurações do Servidor

Hardware/Software	Descrição
Processador	Intel Core 2 Duo E7500 @2.93
Memória RAM	4GB RAM DDR2 1066Mhz
Espaço em HD	WDC WD3200AAJS-00L7A0 SATA
Sistema Operacional (64 bits)	Windows 7 Enterprise SP 1

Todo padrão de comunicação por todo esse sistema é realizado com a estrutura de dados JSON, portanto, fora necessário acrescentar aos dados JSON climatológicos também um objeto chamado "envelope" contendo o comando a ser executado e elementos relacionados a cada comando. A figura 4 demonstra a estrutura que compõe o objeto envelope para executar a função de verificar cidades disponíveis para inclusão.

A sequência de atividades do sistema como um todo até a execução do armazenamento de uma precisão é descrita no diagrama de atividades na figura 1.

Figura 1. Sequência de Execução do Sistema Proposto



Resultados e Discussões

Para os testes a massa de dados de previsões usada nos bancos de dados, tanto no MongoDB como no MSSQL, foram 1.007.643 e 970.832 respectivamente.

Para utilizar a interface de conexão do Node.js com o banco de dados MS SQL Server foi instalado o driver padrão node - mssql usando o comando nvm install mssql no terminal do Node.js. Após a instalação podemos carregar o modulo mssql com a

função require(), e a partir do driver acessar as funções de conexão, de execução de querys, entre outros.

O módulo para o acesso ao MongoDB já é nativamente instalado no Node.js, usamos o módulo mongose para a definição das estruturas JSON para o MongoDB e o mesmo para as funções de manipulação.

Nossa rotina de teste pode ser resumida em execução durante um período de 4 horas de inserções e previsões para cada banco.

Foi realizado o teste em ambos os bancos de dados durante um período de 4h, analisando o tempo necessário para cada execução e o consumo de recursos do processador. As funções de análise de tempo são descritas como:

$$QOP = \sum_{i=1}^{n} OPi + 1$$

A função apresentada acima foi definida para descrever a quantidades de execuções de um determinado tipo de operação (Municípios disponíveis, Alteração dos Municípios disponíveis para verificado, Armazenamento de Previsão).

TOP = Tempo total de execução de operação em algum momento.

$$STOP = \sum_{i=1}^{QOP} TOPi$$

A função acima descreve a somatória do total de tempo necessário para executar todas as operações de um tipo. STOP = Somatória dos Tempos totais de operação.

$$MTOP = \frac{STOP}{QOP}$$

A função identificada acima apresenta a média de tempo necessário de execução de um tipo de operação. MTOP = Média em milissegundos de execução de operação. Sobre resultados descritos a seguir devemos desconsiderar os itens c) e d) por realizar suas operações em uma massa de 6335 cidades.

Os resultados obtidos na primeira bateria de testes para o MongoDB, foram:

a) Algo a se ressaltar no resultado da execução dos testes é o grande consumo de recursos do MongoDB, ele suportou apenas 4 instancias de aplicações cliente,

consumindo entre 78% e 100% com constantes picos máximos. Isto se deve ao motor de armazenamento WiredTiger que comprime o armazenamento de dados ao fim de cada operação e descomprime temporariamente quando necessário. Este processo não geraria consumo de recursos além do necessário para a operação caso não houvesse a fase de compressão/descompressão. Algumas soluções para melhorar esse consumo de CPU são aumentar a capacidade de memórias RAM para que todo o processo execute nesta memória, que é mais rápida comparado ao HD, outra solução seria incluir novos nós com mais memória ou comprimir os índices. Comprimir os índices pode reduzir os dados comprimidos em cinco a dez vezes o tamanho total para compressão resultando em menos consumo de recurso de processamento e memória RAM. O WiredTiger está ativo por padrão desde a versão 3.0, versões anteriores ativavam por padrão o motor MMAP, este motor não realizava compressão, por tanto, os dados armazenamos possuíam tamanhos gigantescos, com o WiredTiger armazenamentos com massas gigantescas são agora otimizadas em relação a uso de memória RAM e armazenamento em HD, porém, o custo médio de processamento por operação aumentou drasticamente;

b) O tempo médio para uma consulta relacionada a data de uma previsão utilizando análise de texto foi de 1440ms;

c) O Resultado para a verificação de cidades disponíveis para inserção de previsão resultou em 340 operações com a média de desempenho de 30 milissegundos;

d) Sobre o resultado da média de operações de atualização de cidade já verificado, obtivemos 84 milissegundos, em um total de 17000 operações;

e) As inserções de novas previsões no MongoDB resultaram em uma média de 1042 milissegundos para executar 13042 operações.

No MSSQL quanto mais caracteres for necessário analisar no parse mais tempo se leva e nesse período de execução o uso de CPU é máximo. O MSSQL suportou vinte instâncias da aplicação cliente, usado aproximadamente 3 a 4% constante da CPU ou 1.6 de impacto médio na CPU em um período de 60s no sistema servidor. Nas aplicações Cliente o uso de recurso foi mínimo, entre 1.5 e 2.9% de uso de CPU. Os resultados podem ser vistos no gráfico 2 e sobre os resultados descritos a seguir

devemos desconsiderar os itens a) e b) por realizar suas operações em uma massa de 6335 cidades:

a) O Resultado da média de consulta de previsões disponíveis foram de 22 milissegundos em 19754 operações;

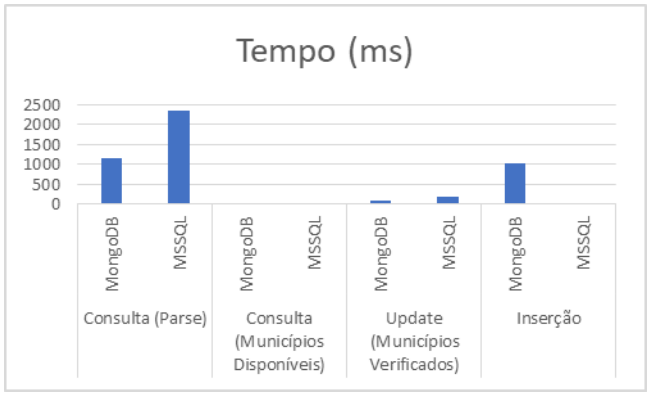
b) O Resultado da média de atualização das previsões disponíveis no MSSQL fora 177 milissegundos em um total de 34300 operações;

c) O Resultado da média de inserção foi de 17 milissegundos em 34300 operações.

d) O produto dos testes em consultas com análise de texto fora de 2342ms.

O gráfico 1 apresenta a comparação da primeira bateria nas execuções em MongoDB e MSSQL.

Gráfico 1. Comparação dos resultados das operações da primeira bateria de testes em MS SQL Server e MongoDB



Os resultados obtidos na segunda bateria de testes para o MongoDB, com 10917675 de coleções, foram:

a) Na primeira bateria de testes do MongoDB o consumo de recursos para o funcionamento do sistema foi alto a ponto de chegar ao consumo máximo e assim fazendo as instâncias cliente disputarem por recurso de CPU resultando em menos operações executadas por hora por causa da compressão realizada pelo WiredTiger, porém as consultas não foram afetadas. Na segunda bateria de testes o consumo foi intensificado tanto nas inserções, o tempo médio de inserções aumentou em 28 vezes,

como nas consultas com análise de texto, nesta bateria de testes as consultas com parse tem seus resultados apresentados de forma a separar consultas com um ano especificado, um ano e um mês especificado, um ano, um mês e um dia especificado, e um ano, um mês, um dia e uma hora especificado;

b) O Resultado para a verificação de cidades disponíveis para inserção de previsão resultou em 50 operações com a média de desempenho de 41 milissegundos;

c) Sobre o resultado da média de operações de atualização de cidade já verificado, obtivemos 111 milissegundos, em um total de 2500 operações;

d) As inserções de novas previsões no MongoDB resultaram em uma média de 29280 milissegundos para executar 1557 operações.

e) Foram realizadas 3 execuções de consulta semelhantes, e é possível notar que a cada execução o tempo necessário é menor que a consulta anterior, com exceção da terceira consulta do gráfico 8 por ter sido incluído mais uma restrição para consultar também a previsões com datas referentes ao mecanismo meteorológico ClimaTempo. Durante os testes notamos que a performance do MongoDB se mostrou otimizada em relação a consulta anterior. Devemos notar que na primeira consulta após a inicialização do MongoDB é necessário a transferência das coleções do disco rígido para a memória RAM e por isso a primeira consulta apresentou um acréscimo considerável no tempo necessário para ser realizada.

Os resultados obtidos na segunda bateria de testes para o MSSQL, com 10917675 de coleções foram:

a) O Resultado da média de consulta de previsões disponíveis foi de 33 milissegundos em 2830 operações, como é demonstrado no gráfico 10;

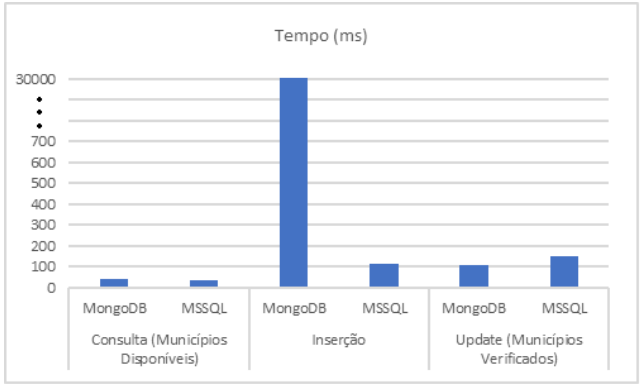
b) O Resultado da média de atualização das previsões disponíveis no MSSQL fora 148 milissegundos em um total de 30500 operações;

c) O Resultado da média de inserção foi de 113 milissegundos em 30500 operações.

d) O produto dos testes em consultas com análise de texto apresentou resultados diferenciados conforme mais dados eram inclusos nas consultas, portanto, os resultados estão divididos em quatro gráficos, cada qual com restrições de consultas diferentes. Conforme visto anteriormente, a primeira consulta demanda demasiado tempo de execução visto que os dados não estão na memória RAM. O MSSQL utiliza algoritmos automaticamente para otimizar as consultas usadas com frequência, nos demais gráficos isto também se repete. Na segunda consulta apresentada no gráfico o tempo para retornar os registros foi de 30,6 segundos e na terceira consulta com um outro ano reduziu-se 5 segundos. Os resultados apresentados mostram valores muito semelhantes nas três execuções com variações leves, os resultados da consulta diária apresentaram grande variação na primeira consulta, mas com a realização de outras *querys* semelhantes o MSSQL otimiza essas operações, como é vista no gráfico 13, e na última consulta o campo mecanismo também está incluído, assim como nas baterias de testes do MongoDB. Por fim a consulta por horário apresentou resultado notável na terceira consulta, reduzindo aproximadamente 4 segundos no tempo de execução comparado as consultas anteriores, e chegando próximo do resultado nesta mesma consulta no MongoDB.

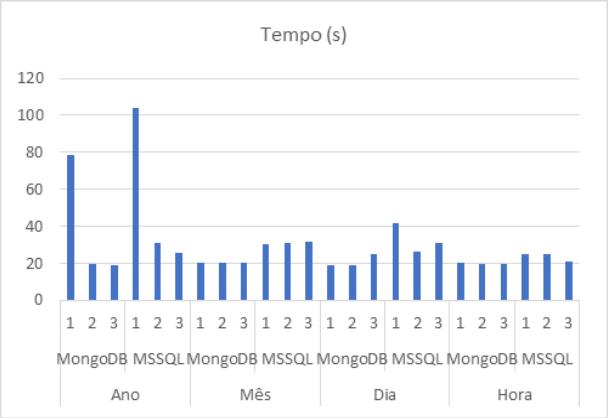
O gráfico 2 apresenta a comparação das medidas na segunda bateria de testes.

Gráfico 2. Comparação dos resultados das operações da segunda bateria de testes em MS SQL Server e MongoDB



O gráfico 3 apresenta as consultas detalhadas na segunda bateria de testes.

Gráfico 3. Comparação dos resultados das operações de consulta na segunda bateria de testes em MS SQL Server e MongoDB



Considerações Finais

A busca por alternativas para persistência de dados cresce a cada dia. A necessidade de alta escalabilidade horizontal fez surgir alternativas aos SGBDs relacionais. Porém, assim como os bancos de dados relacionais não se aplicam a todas as situações, o mesmo ocorre com os bancos NoSql.

A partir dos resultados obtidos em nossa pesquisa exploratória pode-se perceber que o uso de ferramentas NoSQL como o MongoDB tem como características:

a) O armazenamento dos dados com a compressão do motor WiredTiger resultou em menos uso de recursos de armazenamento, em massas de dados gigantescas esse benefício é mais significativo;

b) As consultas com análise de texto são claramente mais rápidas independente do texto analisado;

c) Uma desvantagem é vista no tempo gasto para inserção de novos documentos, devido a compressão realizada;

Os resultados obtidos também demonstraram pontos de interesse do desempenho do SQL Server:

a) Algo que se destaca no modelo relacional é o seu desempenho na inclusão de novos registros, muitíssimo superior ao MongoDB, esperávamos um melhor desempenho, mas, os valores nos surpreenderam;

b) A consulta com análise de texto mostrou ser inferior entre 20% a 40% em tempo necessário para execução;

Ao comparamos os dois bancos de dados foi possível confirmar que em grandes massas de dados a realização de consultas no MongoDB são significativamente melhores, e apesar da inserção ser prejudicada, as vantagens que esse banco apresenta o tornam viável caso o sistema envolvido tenham um alto grau de consultas ao banco de dados. Também a possibilidade de flexibilidade dos documentos se mostra vantajoso ao modelo relacional. Por outro lado, notamos que o SQL Server consome menos recursos em operações gerais e, portanto, inserir dados se torna mais rápido, permitindo responder a uma alta quantidade de requisições. Caso a necessidade de escalamento vertical seja possível, a estrutura de dados esteja definida, não seja complexa e não tenha uma grande previsão de crescimento.

Referências

TILKOV, Stefan; VINOSKI, Steve. Node. js: Using JavaScript to build high-performance network programs. IEEE Internet Computing, v. 14, n. 6, p. 80-83, 2010

CURBERA, Francisco et al. Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. IEEE Internet computing, v. 6, n. 2, p. 86-93, 2002.

CHAPPLE, Mike. Microsoft SQL Server 2008 para Leigos. Rio de Janeiro. Alta Books. 2009.

IZQUIERDO, Javier Luis Cánovas; CABOT, Jordi. Discovering implicit schemas in JSON data. In: International Conference on Web Engineering. Springer, Berlin, Heidelberg, 2013. p. 68-83.

DATE, Christopher John. An introduction to database systems. Pearson Education India, 2006.

IZQUIERDO, Javier Luis Cánovas; CABOT, Jordi. Discovering implicit schemas in JSON data. In: International Conference on Web Engineering. Springer, Berlin, Heidelberg, 2013. p. 68-83.

TIWARI, Shashank. Professional NoSQL. John Wiley & Sons, 2011.

CHODOROW, Kristina. MongoDB: The Definitive Guide: Powerful and Scalable Data Storage. " O'Reilly Media, Inc.", 2013.

DAYLEY, Brad. Node. js, MongoDB, and AngularJS web development. Addison-Wesley Professional, 2014.

MONGODB. Journaling. set. 2016. Disponível em: <https://docs.mongodb.com/manual/core/journaling/>. Acesso em: 5 nov. 2016.

MONGODB. Map-Reduce. set. 2016. Disponível em: <https://docs.mongodb.com/v3.2/core/map-reduce/>. Acesso em: 5 nov. 2016.

MONGODB. Sharding. set. 2016. Disponível em: <https://docs.mongodb.com/manual/sharding/>. Acesso em: 5 nov. 2016.

ZUR MUEHLEN, Michael; NICKERSON, Jeffrey V.; SWENSON, Keith D. Developing web services choreography standards—the case of REST vs. SOAP. Decision Support Systems, v. 40, n. 1, p. 9-29, 2005.

GUIMARAES, Celio Cardoso. Fundamentos de bancos de dados: modelagem, projeto de linguagem SQL. Ed. da Unicamp, 2003.

NAYAK, Ameya; PORIYA, Anil; POOJARY, Dikshay. Type of NOSQL databases and its comparison with relational databases. International Journal of Applied Information Systems, v. 5, n. 4, p. 16-19, 2013.

AHO, Alfred V.; SETHI, Ravi; ULLMAN, Jeffrey D. Compilers, Principles, Techniques. Addison Wesley, v. 7, n. 8, p. 9, 1986.

SILBERSCHATZ, Abraham; KORTH, Henry; SUNDARSHAN, S. Sistema de banco de dados. Elsevier Brasil, 2016.

CATTELL, Rick. Scalable SQL and NoSQL data stores. Acm Sigmod Record, v. 39, n. 4, p. 12-27, 2011.