# Performace de Query e Escalabilidade no NoSQL: MongoDB vs CouchDB

**Allison Alfredo de Oliveira Sampaio**[1], **Mara Luci Leite Goulart**[2]

[1]Departamento de Ciência da Computação – Universidade Tecnológica Federal do Paraná (UTFPR)
R. Rosalina Maria Ferreira, 1233 – Vila Carola, Campo Mourão – PR, 87301-899

[2]Departamento de Ciência da Computação – Universidade Tecnológica Federal do Paraná
Campo Mourão, PR, Brasil

allisonsampaiox@gmail.com, maragoulart@alunos.utfpr.edu.br

***Abstract.*** *NoSQL databases are becoming increasingly popular and used in recent years, due to the need to store large amounts of data that organizations have to work with, as well as the increase in users of systems. Traditionally used databases, such as relational databases, limiting factors due to the fact that they do not present much flexibility in the structuring of data. With this, the NoSQL Databases are an alternative that to address these and other shortcomings, such as scalability issues and system availability. That said, you need to know which database category NoSql best applies to each project. A strength of this technology is the ability to replicate. Thus, the objective of this work is to perform a comparison between MongoDB and CouchDB testing this aspect of both technologies.*

***Resumo.*** *Os bancos de dados NoSQL estão tornando-se cada vez mais conhecidos e utilizados nos últimos anos, em razão da necessidade de armazenamento de grande quantidade de dados com os quais as organizações têm que trabalhar e também do aumento de usuários dos sistemas. Os bancos de dados tradicionalmente utilizados, como os relacionais, apresentam fatores limitantes devido ao fato de não apresentarem muita flexibilidade na estruturação dos dados. Com isto, os Bancos de Dados NoSQL são uma alternativa que objetiva suprir estas e outras deficiências, como as questões de escalabilidade e disponibilidade do sistema. Dito isso é preciso saber qual categoria de banco de dados NoSql se aplica melhor a cada projeto. Um ponto forte dessa tecnologia é a capacidade de replicação. Assim, o objetivo deste trabalho é realizar uma comparação ente o MongoDB e o CouchDB testando esse aspecto de ambas as tecnologias.*

## 1. Introdução

Durante um longo período o modelo de dados relacional tem sido amplamente utilizado pela maioria dos Sistemas de Gerenciamento de Banco de Dados (SGBD). Ainda hoje é assim. Entretanto, nos últimos anos vem ocorrendo um crescimento significativo de aplicações, recursos web e tudo que se refere a sistemas computacionais. Essas aplicações geram grande volume de dados e têm o desafio de servir uma grande quantidade de usuários, que esperam o perfeito funcionamento dos sistemas. Nesse cenário, a utilização do modelo relacional não se mostra tão eficiente, uma vez que este modelo

possui sérias limitações quanto à escalabilidade horizontal, já que não foi projetado inicialmente para tal. Como organizar os dados de um Sistema de Gerenciamento de Banco de Dados Relacional (SGBDR) em sistemas distribuídos é complexo, quase sempre se acaba recorrendo à escalabilidade vertical para aumento de desempenho, ou seja, faz-se upgrade dos sistemas de hardware do servidor. Para resolver esse problema surgem os bancos de dados NoSQL, que possuem diversas categorias, para o presente estudo foi escolhida a categoria de bancos de dados orientado a documentos, dois clientes conhecidos são o MongoDB e o CouchDB. O CouchDB é um banco de dados de documentos flexível, que suporta JSON e preza pela consistência de dados. O MongoDB é um banco de dados não relacional, e utiliza BJSON para armazenar dados. Possui um significativo suporte de query e índices, o que permite um acesso rápido a grandes plataformas de dados. Durante o trabalho foi proposto um teste de replicação, que consiste em um teste de stress, no qual simulou-se o acesso simultâneo de vários usuários a um banco de dados, buscando um comparativo de desempenho entre o MongoDB e o CouchDB. O trabalho apresenta o seguinte conteúdo: a) Objetivo; b) Conceitos; c) Métodos de avaliação; d) Resultados e) Conclusão·

## 2. Objetivo

O objetivo do trabalho é realizar uma pesquisa com banco de dados NoSQL pré-selecionados para estudo e comparação, selecionando os bancos estudados para o desenvolvimento de uma aplicação para manipular dados armazenados nestes bancos.

## 3. Conceitos

Em meados dos anos 60 foi introduzido o termo banco de dados que serviria para armazenar informações, uma camada de suporte para os sistemas de informações. O conceito de separar aplicações dos dados era novo e abria possibilidades para aumentar a robustez das aplicações. Em 1970 Edgar Codd propos o modelo relacional que substituiu os modelos hierárquicos e de rede da época, o modelo de Codd tornou-se base para o uso de SQL que permite que os dados sejam armazenados em tabelas. Por volta dos anos 2000, as aplicações começaram a produzir um vasto volume de dados por meio de de aplicações complexas. Segundo [Juravich 2012] , termo NoSQL não é a abreviação de “no SQL” – sem SQL, significa “not only SQL”, não apenas SQL, bancos de dados NoSQL são soluções de persistência de dados que não seguem o modelo relacional e não utiliza SQL para querying. A primeira característica que faz os bancos de dados NoSQL diferente é a sua estrutura de dados. Dentre as estruturas disponíveis, há a orientada a documentos, que salva os itens sem serem estruturados em um esquema. Essa estrutura de dados é remanescente de um array associativo em PHP. [Juravich 2012] acrescenta que o NoSQL deve ser utilizado quando o esquema e a estrutura dos dados mudam constantemente.

### 3.1. CouchDB

O CouchDB utiliza uma API HTTP/REST para a manipulação dos dados, portanto a comunicação entre entre cliente e servidor pode ser realizada em qualquer aplicação que possa comunicar-se por HTTP. As operações básicas de um banco de dados são mapeadas para as operações do protocolo HTTP, como GET, PUT, POS T, DELETE, para ler, armazenar, substituir e remover objetos armazenados [Juravich 2012]. Além da API REST/HTTP que provê acesso a qualquer aplicação com suporte ao protocolo HTTP, o

CouchDB oferece um servidor web, chamado Futon, que pode ser acessado pelo browser através do endereço /localhost 5984/ utils/, onde é possível gerenciar os bancos e documentos individualmente. A arquitetura do CouchDB consiste em um servidor de banco de dados master que possui um ou mais bancos de dados criados. Os bancos são compostos por objetos, que são documentos do tipo JSON, de esquema livre, ou seja, em uma mesma base de dados os documento não precisam ter os mesmos campos. Entretanto, recomenda-se armazenar documentos semelhantes em um mesmo banco de dados [Redmond and Wilson 2012]. CouchDB suporta múltiplas réplicas do banco de dados rodando em diferentes servidores e fornece um mecanismo para sincronização destes dados. Há um processo, replicador, executando que possibilita informar a origem e o destino do servidor, é realizada uma verificação pela data e caso houve mudanças, a cópia é sobrescrito no servidor slave. As replicações são realizadas de forma assíncrona, de forma que não compromete a velocidade de escrita e leitura [Juravich 2012]. Dentre as principais características do CouchDB, estão:

**a** ) Um servidor de banco de dados de documento, acessível via RESTful JSON API.
**b** ) Ad-hoc e esquema com um largo espaço de endereço.
**c** ) Queries feitas a partir do JavaScript.

### 3.2. MongoDB

O aplicativo MongoDB é composto por dois tipos de serviços, o processo servidor mongod que é o núcleo do banco de dados e o serviço mongos para autosharding. Sharding é a divisão dos dados em vários nós, sendo utilizado quando faz-se necessário balanceamento de carga . O processo servidor pode rodar tanto em arquiteturas 32 como 64- bit, no entanto, é recomendado o uso de 64-bit, uma vez que o tamanho máximo de um banco de dados é limitado à 2GB no modo 32-bit. Além dos serviços do aplicativo servidor, há também serviço cliente mongo, que é um cliente shell padrão do MongoDB utilizado para conectar-se ao servidor através da porta 27017. Entretanto, há uma grande variedade de drivers oficiais disponíveis, como C, C++, Haskell, Java, PHP, entre outros, todos eles estão sob a licença Apache [Redmond and Wilson 2012]. Também é possível conectar através da interface HTTP/REST na porta 28017, permitindo a manipulação de entradas via HTTP. O servidor MongoDB pode hospedar mais de um banco de dados, independentes entre si, armazenados separadamente. Um banco de dados contém um uma ou mais coleções constituídas por documentos BSON (Binary JSON), que são estruturados como documentos JSON, com esquema dinâmico, fazendo com que a integração de dados em certos tipos de aplicações sejam mais fáceis e rápidos [Garret 2016]. Os autores acrescentam que consultas são expressas em sintaxe como JSON e são enviadas ao servidor como objetos BSON pelo driver do banco de dados. O modelo permite consultas a todos os documentos dentro de uma coleção, incluindo objetos e matrizes incorporadas. MongoDB não possui transações, tampouco joins, ficando a cargo do desenvolvedor implementá-las, se necessário, em uma aplicação. É possível inclusive relacionar documentos de coleções distintas. Quanto aos recursos disponíveis, vale ressaltar o suporte automático para a realização de Sharding. O sistema de banco de dados pode ser distribuído através de um cluster de máquinas. O cluster consiste em três componentes, nós Shard, servidores de configuração e serviços de roteamento chamados mongos. Os nós Shard são responsáveis por armazenar os dados atuais, podendo inclusive ser replicados para redundância em caso de falha. Os servidores de configuração são usados para armazenar os metadados e rotear

informação do cluster de MongoDB. Os Mongos são responsáveis por rotear os processos, otimizando o desempenho das tarefas requisitadas pelos clientes [Garret 2016]. Outro recurso importante é o GridFS, que serve para armazenar e recuperar arquivos que excedam o limite de 16M dos documentos BSON. Os arquivos são divididos em partes e armazenados em uma coleção, em outra coleção são armazenados os metadados do arquivo. É útil para armazenar arquivos grandes, como áudio e vídeo [Redmond and Wilson 2012].

## 4. Método de Avaliação

O método de avaliação do presente trabalho, será feito por meio do desenvolvimento de uma aplicação web que armazenará um pacote json de 74.5MB contendo 18.000 arquivos json. Os bancos de dados NoSQL utilizam queries dinâmicas, por exemplo, uma query SQL:

```
SELECT * FROM posts
INNER JOIN posts_tags ON posts.id = posts_tags.post_id
INNER JOIN tags ON posts_tags.tag_id == tags.id
WHERE tags.text = 'politics' AND posts.vote_count > 10;
```

E uma query equivalente em MongoDB

```
db.posts.find({'tags': 'politics', 'vote_count': {'$gt': 10}});
```

Aplicaremos um teste de stress de banco de dados quanto a replicação, partindo do princípio do que faz os bancos de dados orientados a documento únicos é a sua capacidade de lidar com documentos sem esquema definido em tabelas como noSQL. Eles possuem a capacidade de replicar suas coleções em diferentes servidores e performar queries em paralelo, se um servidor não está disponível ele faz requisição para o próximo [Redmond and Wilson 2012]. Os autores acrescentam que no caso do CouchDB ele permite múltiplos mestres, já o Mongo considera como sua força principal a abilidade de lidar com um grande volume de dados e múltiplas requisições por meio de replicação e escala horizontal, uma vez que o bancos de dados relacionais utilizam escala vertical.

### 4.1. Requerimentos do Sistema

O ambiente utilizado para a realização dos testes de desempenho possui um processador Intel Core i3 de sexta geração, sendo a memória de 4GB e o sistema operacional Windows 10 64 bits.

- Notebook Inspiron 15 5000 i15-5566-A10P
- Sistema Operacional: Windows 10 Home Single Language x64
- Processador: Intel(R) Core(TM) i3-6006U CPU @ 2.00GHz
- Memória (RAM): 4,00GB

### 4.2. Instalação MongoDB e CouchDB

O MongoDB é autônomo e não possui outras dependências do sistema. Você pode executar o MongoDB a partir de qualquer escolhida conforme o tutorial de instalação da própria aplicação [MongoDB 2017]. Quanto ao CouchDB não há qualquer pré-requisito extra para a instalação, sendo clara a documentação apresentada no próprio site [CouchDB 2017].

## 5. Resultados

Para a execução do teste de stress de banco de dados, utilizamos a ferramenta jmitter. O JMeter é uma aplicação implementada em Java projetada para fazer teste de carga de aplicações Cliente/Servidor. Foi originalmente desenhado para realizar testes em aplicações Web, mas, logo foi expandido para fazer outros tipos de testes, tais como em: servidores de banco de dados . Com o JMeter, é possível assegurar se um SGBD está capacitado para suportar uma determinada quantidade de usuários simultâneos e estudar o seu comportamento no que tange à escalabilidade. Neste contexto, entende-se por escalabilidade a capacidade de resposta do sistema em relação à demanda de recursos exigidos dele. Foi utilizada a replicação passiva, ou seja ter um master enviando todas as requisicões e os demais servidores as recebendo. O CouchDB permite a utilização de múltiplos mestres, o que torna algumas réplicas mestres, no entanto, essa configuração não será abordada neste trabalho. O teste escolhido JMeter Distributed Test, que consiste em configurar um servidor mestre enviando requisições para outros slaves, conforme a configuração abaixo. IP's:

```
172.0.0.1, 172.0.0.2, 172.0.0.3, 172.0.0.4, 172.0.0.5, 172.0.0.6
```

Se for necessário mais máquinas, basta indicar mais números de IP's. Foram selecionados três cenários com 5, 8 e 10 máquinas de replicação. Os resultados estão apresentados nas tabelas a seguir:

A Tabela 1 apresenta o resultado do teste de replicação para o MongoDB e o CouchDB no cenário 1:

**Tabela 1. Cenário 1**

| | Quantidade json | Número de replicas | Tempo em segundos |
|---|---|---|---|
| **MongoDB** | 95000 | 5 | 9000 |
| **CouchDB** | 95000 | 5 | 7200 |

A Tabela 2 apresenta o resultado do teste de replicação para o MongoDB e o CouchDB no cenário 2:

**Tabela 2. Cenário 2**

| | Quantidade json | Número de replicas | Tempo em segundos |
|---|---|---|---|
| **MongoDB** | 120000 | 8 | 14400 |
| **CouchDB** | 120000 | 8 | 13750 |

A Tabela 3 apresenta o resultado do teste de replicação para o MongoDB e o CouchDB no cenário 3:

**Tabela 3. Cenário 3**

| | Quantidade json | número de replicas | Tempo em segundos |
|---|---|---|---|
| **MongoDB** | 250000 | 10 | 18000 |
| **CouchDB** | 250000 | 10 | 16200 |

Para uma visão geral dos resultados obtidos foi gerado um gráfico de linha que compara o tempo medido em segundos pelo total de dados replicados em cada máquina. A execução ocorre em três cenários descritos a seguir:

**Cenário 1:** Entrada de 95000 arquivos JSON replicados em 5 máquinas
**Cenário 2:** Entrada de 120000 arquivos JSON replicados em 8 máquinas
**Cenário 3:** Entrada de 250000 arquivos JSON replicados em 10 máquinas

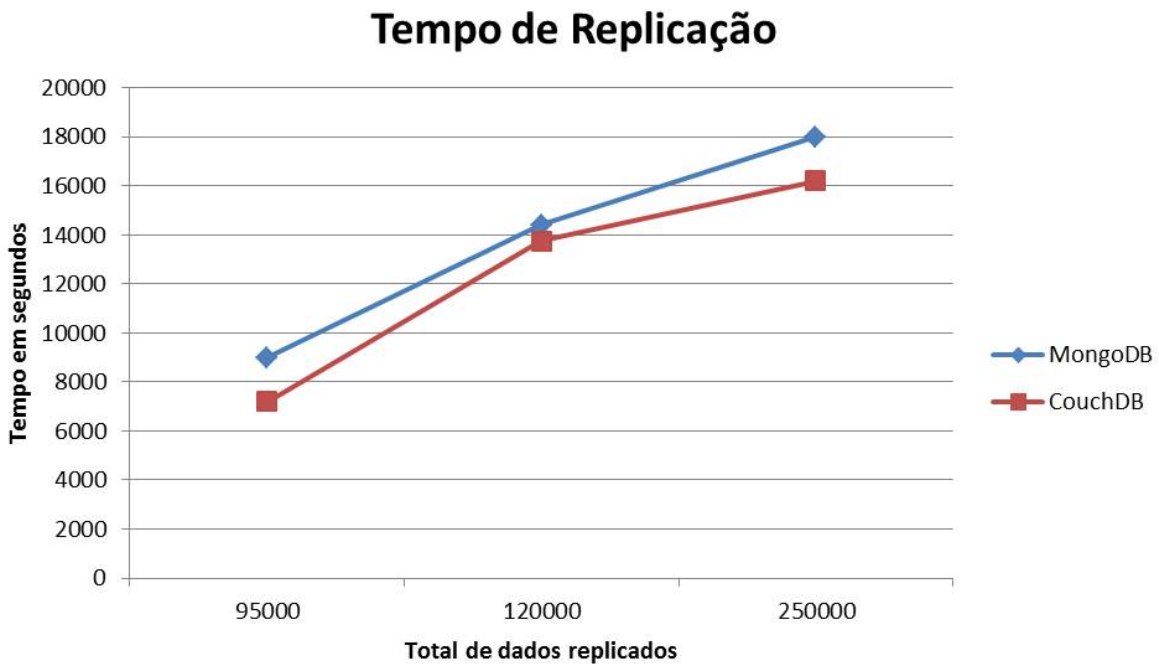


**Figura 1. Comparação de tempo de replicação do MongoDB e CouchDB**

Como observado no gráfico, percebe-se que o CouchDB leva menos tempo para replicar os dados nos três cenários, mesmo variando o número de máquinas.

## 6. Conclusão

A finalidade deste trabalho foi fornecer uma visão geral e introduzir os conceitos de bancos de dados NoSQL , que surgiu como uma alternativa aos SGBD predominantemente relacionais. Para isto, um dos objetivos foi analisar dois bancos de dados NoSQL disponíveis para utilização e que já tinham uma certa popularidade. Foram analisados os bancos CouchDB e MongoDB. O estudo desenvolvido sobre os banco de dados NoSQL teve a intenção de adquirir conhecimentos sobre como cada um deles funciona, de que forma os dados são armazenados, qual a arquitetura que cada um deles utiliza, quais os requisitos para o funcionamento, e principalmente o seu comportamento durante um teste de stress, para medir a eficiência da funcionalidade de replicação, algo que diferencia esses bancos de dados dos bancos de dados relacionais. Ao final do teste de stress nota-se um desempenho superior do banco de dados CouchDB em relação ao MongoDB, esse resultado pode ser creditado ao fato do CouchDB permitir múltiplos mestres, enquanto o MongoDB permite apenas um.

## Referências


CouchDB (2017). *Couch Documentation*. CouchDB.

Garret, D. (2016). *Mongo in Action*. Manning Publications, 1th edition.

Juravich, T. (2012). *CouchDB and PHP Web Development*. Packt Publishing, 1th edition.

MongoDB (2017). *Mongo Documentation*. MongoDB.

Redmond, E. and Wilson, J. R. (2012). *Seven Databases in Seven Weeks*. LLC, 1th edition.