Amadurecimento, Consolidação e Performance de SGBDs NoSQL - Estudo Comparativo

Alternative Title: Maturing, Consolidation and Performance of NoSQL Databases - Comparative Study

Vanessa Cristina Oliveira de Souza
Universidade Federal de Itajubá
Instituto de Matemática e Computação
Caixa Postal: 50 - CEP: 37500 903 - Itajubá – MG
55(35)3629-1827
vanessasouza@unifei.edu.br

Marcus Vinícius Carli dos Santos
Universidade Federal de Itajubá
Instituto de Matemática e Computação
Caixa Postal: 50 - CEP: 37500 903 - Itajubá – MG
55(35)3629-1827
marcusdcarli@gmail.com

RESUMO

Presencia-se ultimamente a expansão no número de dados gerados nas mais diversas áreas da computação. O modelo tradicional de banco de dados, chamado relacional não consegue lidar com este crescimento. O NoSQL surge como uma solução para esta problemática. Este trabalho teve por finalidade realizar um estudo sobre a arquitetura de alguns SGBDs NoSQL, evidenciar o motivo pelo qual eles conseguem lidar com grandes volumes de dados e mostrar algumas propriedades que o NoSQL baseia-se para fazer a gestão de dados. Para tanto, os SGBDs Redis e Cassandra foram utilizados e comparados ao banco relacional MySQL. Os resultados corroboram a literatura. O Redis apresentou melhor performance e o Cassandra a melhor escalabilidade. A maturidade e consolidação dos SGBDs NoSQL foram qualitativamente avaliados, de modo que este trabalho apresenta uma percepção de um usuário já familiarizado com a abordagem relacional dando seus primeiros passos com o NoSQL.

Palavras-Chave

NoSQL, escalabilidade, performance, consolidação

ABSTRACT

This paper presents a study on the NoSQL (Not Only SQL), a new dimension in databases. Lately witnesses an expansion in the number of generated data, and the relational model cannot deal with this data growth. The NoSQL comes as a solution to this problem. This paper aims to conduct a study on the architecture of DBMS, demonstrate why they can deal with this large volume of data and show some properties that NoSQL is based to make your data management. Therefore, DBMSs Redis and Cassandra were used and compared to the relational database MySQL. The maturity and consolidation of NoSQL DBMSs were qualitatively assessed on the parameters online documentation, software help, support virtual community and academic articles, so this work presents a perception of a user already familiar with relational approach taking their first steps with NoSQL. The results showed recognition of the relational database, but also showed a strong consolidation of NoSQL technology market and academic research environments, especially with Cassandra. The horizontal scale was tested with the DBMS Cassandra, who proved to be an excellent tool in this regard.

SBSI 2015, May 26–29, 2015, Goiânia, Goiás, Brazil.

Categories and Subject Descriptors

H.2.4 [**Database Systems]**

General Terms

Performance, Design, Experimentation

Keywords

NoSQL, performance, horinzontal scale, consolidation

1. INTRODUÇÃO

Com a expansão da Internet, o volume de dados cresce de forma exponencial e, nesse contexto, o *Big Data* vem ganhando grande importância. Em aplicações como redes sociais e sites de e-commerce, com inúmeros acessos e transações diárias, os sistemas de banco de dados relacionais tendem a ficar sobrecarregados. No modelo relacional[1, 2] os dados são estruturados, armazenados, manipulados e recuperados em forma de tabelas normalizadas, e todas as transações seguem propriedades de forte consistência chamadas ACID (Atomicidade, Consistência, Isolamento, Durabilidade).

Apesar da riqueza de recursos, os SGBDs relacionais tendem a aumentar a complexidade de utilização com o aumento do fluxo de dados. Eles não foram criados para trabalhar em ambientes distribuídos e o escalonamento, em geral, se dá pelo aumento da capacidade da máquina servidora (escalonamento vertical). Acontece que ao aumentar muito o fluxo de dados, o uso de *cluster*s torna-se inevitável e o desempenho dos SGBDs relacionais cai.

Para suportar aplicações *Big Data*, onde é grande o fluxo de dados e a necessidade de escalabilidade dos servidores, surgiram os bancos de dados não relacionais, chamados NoSQL (*NotOnly SQL*), que apresentam-se como uma otimizada solução onde os bancos relacionais têm grande deficiência.

De fato, o desenvolvimento da internet e da computação em nuvem fez surgir novos tipos de aplicações que destacou a tecnologia de banco de dados NoSQL. Aspectos como: escrita e leitura de alta concorrência com baixa latência, armazenamento e acesso eficiente de *Big Data*, alta escalabilidade e alta disponibilidade, baixo custo de administração e operação, apontam bancos NoSQL como uma forma eficiente e de baixo custo para suprir essa demanda.

[3] destacam a utilização de bancos NoSQL na computação em nuvem (*cloudcomputing*) e por empresas como IBM, Twiter, Facebook, Google e Yahoo!, com a finalidade de processamento analítico de *logs* Web, transações convencionais, entre outras tarefas. Bancos de dados NoSQL estão sendo adotados também em instituições financeiras, agências governamentais, comércio de produtos de varejo, entre outras áreas onde bancos de dados relacionais eram dominantes. Para se ter uma ideia de sua popularidade, o *ranking* dos sistemas gerenciadores de banco de dados mostra que bancos NoSQL já aparecerem entre os dez mais utilizados no mercado [4].

É importante ressaltar que o *Big Data* foi um estopim para a ascensão dos bancos NoSQL, mas como cita [5], ele não é o único motivo. A facilidade do uso de bancos NoSQL em ambientes distribuídos pode ser, entre outros, um motivo para a escolha desse tipo de *software*.

Dado esse contexto, o objetivo desse trabalho foi comparar três SGBDs, sendo dois NoSQL e um relacional. Como a tecnologia é recente, a intenção foi não apenas realizar testes de performance e escalabilidade, mas também verificar o amadurecimento e consolidação dos *softwares*, por meio de uma análise qualitativa de características como documentação, comunidades virtuais e artigos científicos. A hipótese testada é de que quanto mais completas e volumosas as características acima descritas, maior o amadurecimento e consolidação do *software* no mercado.

2. ESTADO DA ARTE

Segundo [6], *Big Data* descreve conjuntos de dados que crescem exponencialmente e que são demasiadamente volumosos, brutos ou desestruturados para serem analisados por meio de técnicas tradicionais de bancos de dados relacionais e de *business intelligence*. A *Big Data* expande-se através de três dimensões [7]:

- Volume – o tamanho dos dados é muito grande (na casa dos terabytes e petabytes).
- Velocidade – a velocidade que estes dados são criados é muito grande.
- Variedade – há uma grande variedade no formato dos dados, que inclui não apenas dados estruturados, mas também dados semi-estruturados e não estruturados de toda a espécie, como texto, áudio, vídeo, *posts*, arquivos de registros.

[8] apontam algumas limitações com relação à utilização dos bancos relacionais para armazenamento de *Big Data*.

- Leitura e escrita lenta - com o aumento do tamanho dos dados, é possível provocar bloqueios e outros problemas de concorrência, o que leva a um rápido declínio na eficiência da leitura e escrita.
- Capacidade limitada - bancos de dados relacionais atuais não suportam um volume grande de dados no motor de busca.
- Dificuldade de expansão - mecanismos de correlação entre múltiplas tabelas, devido ao uso de chaves estrangeiras existentes em bancos de dados relacionais, não oferecem um bom mecanismo para a escalabilidade, sendo este um fator crucial para atender aos requisitos da *Big Data*.

De fato, [9] aponta a escalabilidade como a principal problemática dos bancos de dados relacionais para lidar com *Big Data*. Escalabilidade é a capacidade do sistema em lidar com o crescimento de dados uniformemente, podendo ser horizontal – que consiste no acréscimo do número de nós em um sistema – e vertical – relacionada à melhoria no hardware presente no sistema. Outra dificuldade inerente aos SGBDs relacionais para lidarem com *Big Data* é que os bancos relacionais são normalizados e seguem as propriedades de consistência ACID (Atomicidade, Consistência, Isolamento e Durabilidade). Somado a isso, há o fato de que, devido à grande quantidade de dados, indexá-los torna-se impraticável, pois o espaço ocupado pelos índices pode chegar a tamanhos críticos, o que leva a uma perda de desempenho.

Sendo assim, para atender a estas limitações, surgiram bancos de dados com arquiteturas e esquemas diferenciados, nomeados NoSQL (*NotOnlySQL*), capazes de processarem dados mais rapidamente, devido aos seus modelos de dados mais simples que os relacionais [8, 10].

Segundo [11], algumas características do NoSQL são: Não utilizam o modelo relacional e nem a linguagem SQL; são projetadas para rodar em *clusters*; tende a ser *open source*; e não possuem um esquema fixo, permitindo a persistência de em qualquer registro.

Os bancos NoSQL seguem a propriedade chamada BASE (*Basically Available, Soft-state, Eventual consistency)*. Segundo [12], o BASE é o oposto de ACID: enquanto o ACID define que ao final de cada operação o banco deve manter sua consistência, o o BASE aceita que a consistência do banco de dados esteja em estado de fluxo, ou seja, tolera inconsistências temporárias para priorizar a disponibilidade.

A disponibilidade básica (*BasicallyAvailable*) determina que o sistema deve estar disponível na maior parte do tempo, mas que subsistemas podem estar temporariamente inoperantes. O estado leve e consistente (*Soft-State*) define que os desenvolvedores devem criar mecanismos para consistências de dados e a consistência eventual (*Eventual consistency*) garante que se certo registro não sofre atualizações, eventualmente estará consistente e todas as leituras retornarão à última atualização do valor. O período entre atualizações e o momento que ela é garantida é chamado de janela de inconsistência [12, 13]. A utilização do modelo BASE possibilita aos SGBDs NoSQL uma flexibilidade que atende as demandas dos ambientes distribuídos.

Uma característica importante de ressaltar dos bancos NoSQL é o teorema CAP [14], base da escalabilidade dos bancos de dados não relacionais. Acrônimo para *Concistency* (Consistência), *Availability* (Disponibilidade), *tolerance 'of network Partition* (tolerância ao particionamento de rede), o teorema descreve que uma aplicação distribuída não pode ter mais que duas propriedades simultaneamente, mas somente duas. Doze anos após a publicação do seu artigo sobre o teorema CAP, [15] percebeu que existem mudanças no cenário atual. Em sistemas onde o particionamento está presente e os projetistas devem fazer escolhas entre disponibilidade e consistência, umas das propriedades não precisa ser necessariamente negligenciada, mas sim apresentar uma combinação dessas. Sistemas NoSQL que fazem uso do particionamento tendem a focar primeiramente em

disponibilidade e deixar a consistência em segundo plano. No entanto, o sistema garante a consistência, mesmo que parcialmente. Hoje em dia, os projetistas devem escolher a que propriedades dar ênfase e não qual propriedade não usar.

Diferente do modelo relacional, os bancos de dados NoSQL possuem diversos modelos de dados que apresentam características que favorecem tipos diferentes de aplicações. Um banco NoSQL pode ser categorizado em quatro modelos diferentes, a saber: chave-valor; orientado a coluna; orientado a documento; e orientado a grafos. Detalhes desses modelos são amplamente encontrados na literatura técnica da área [8, 16, 17, 18, 19, 20, 21], entre outros.

Segundo [22], a utilização de bancos de dados NoSQL é uma alternativa a bancos de dados relacionais, pois provê consultas mais eficientes enquanto mantém flexibilidade e escalabilidade. Bancos de dados NoSQL tem a vantagem sobre a abordagem XML (*eXtensibleMarkupLanguage*) em relação a capacidade de fazer consultas, mas o XML tem a vantagem de reduzir o tempo de preparação para alimentar um banco de dados, principalmente quando é utilizado um banco XML nativo. O NoSQL também partilha desta característica, mas não é tão intuitivo quanto a abordagem XML.

Para [10], daqui para frente o foco do desenvolvimento NoSQL será em melhorar a compatibilidade com aplicações e com ferramentas de gerenciamento. Porém, a adoção do NoSQL em alguns casos será pequena, pois bancos de dados relacionais têm mais tempo de mercado e utilização, além de representarem grandes investimentos por parte dos vendedores e usuários.

3. METODOLOGIA

Este trabalho teve por finalidade realizar testes de performance em dois bancos de dados NoSQL e em um relacional, e revelar as impressões obtidas sobre a maturidade e consolidação dos *softwares*.

3.1 Material

Os SGBDs escolhidos neste trabalho foram o MySQL, Redis e o Cassandra. Utilizou-se também o *benchmark* YCSB. Para a definição dos SGBDs utilizados, levou-se em consideração o ranking dos SGBDS [4], que apontou o Redis e o Cassandra como os SGBDs chave-valor e orientado a coluna mais utilizados. A escolha do MySQL foi devido a sua popularidade e a presença de literaturas que o utilizam em diversas comparações com bancos NoSQL.

O Redis é um banco de dados chave-valor, *open source*, com a característica de que todas as suas operações são feitas em memória e periodicamente os dados são salvos em disco. Segundo [8], ao trabalhar na memória, o Redis garante uma boa performance e pode lidar com mais de cem mil operações de escrita e leitura por segundo. Porém, trabalhar em memória pode ser também uma desvantagem, por estar limitado pela memória do sistema. Segundo [23], o Redis não suporta indexação ou consultas complexas. Para operações básicas é muito rápido e simples, sendo o mais rápido dentre os bancos de dados chave-valor. Sua velocidade e estrutura de dados é uma combinação interessante que pode ser utilizada para a persistência de objetos simples. Estruturas de dados no Redis são chamadas de *Redis Datatypes*. Estas incluem *strings, lists, sets, sorted sets* e *hashes*. Segundo o modelo CAP, o Redis utiliza consistência e tolerância ao particionamento de rede.

Já o Cassandra é um banco de dados orientado a coluna, cujas características mais acentuadas são a escalabilidade e disponibilidade, sem a perda de performance. Foi criado pelo Facebook® e disponibilizado para comunidade em 2008. Ele possui esquema flexível, suporta consultas em range e possui alta escalabilidade [8]. Segundo [24], seu funcionamento consiste em um conjunto de um conjunto de colunas, chamado "supercoluna", semelhante ao modelo chave-valor, no qual no campo valor pode existir outra chave que aponta para outra coluna. Atualizações são armazenadas em memória e posteriormente são escritas em disco, e as informações no disco são periodicamente compactadas. O Cassandra faz particionamento, replicação, detecção de falhas e recuperação automaticamente. No entanto, as cópias dos dados em outros nós são atualizadas assincronamente.

Segundo [25], o particionamento é uma característica chave do Cassandra. Um *cluster* com o Cassandra utiliza o mecanismo de *Gossip*, o que possibilita o controle de estados dos sistemas relacionados. A persistência de dados é feita em disco, todas as escritas em disco são sequenciais e um índice é gerado para buscas mais eficientes. Uma chave em uma família de colunas pode conter muitas colunas. Uma indexação especial é necessária para retornar colunas que estão muito longe da chave. Para evitar que se procure em todas as colunas, um índice de colunas é mantido no disco, o que permite um salto ao setor correto para reaver a coluna [25]. Em relação ao modelo CAP, o Cassandra utiliza disponibilidade e tolerância ao particionamento de rede.

Por sua vez, o MySQL é um banco de dados relacional que utiliza a linguagem SQL (*Structured Query Language*) e apresenta como pontos fortes a facilidade de manuseio, compatibilidade com muitas linguagens e suporte a praticamente todas as plataformas atuais.

Para os testes, fez-se uso do Cassandra versão 1. 12, Redis 2. 6. 7 e MySQL 5. 5. 32. Todos os sistemas foram testados utilizando a configuração padrão, instalados no sistema Linux Mint 15 64 bits. Para o Redis não foi colocado nenhuma restrição de utilização de memória RAM, ou seja, ele pode utilizar toda a memória disponível no sistema.

Segundo [26], *benchmarks* de um SGBD tendem a simular cargas de uma aplicação e gerar relatórios para uma comparação posterior. Esses *benchmarks* criam informações na aplicação que estão medindo o desempenho e simulam o acesso de usuários a esta aplicação. [27] definem o YCSB (*Yahoo! CloudServing Benchmark*) como um framework capaz de realizar *benchmarks* em diferentes sistemas gerenciadores de banco de dados, consistindo em um cliente gerador de cargas de trabalho com diferentes características, tais como cargas de trabalho com muitas leituras, muitas escritas e muitos escaneamentos. Uma das grandes vantagens do YCSB é a capacidade de criar ou adaptar cargas de trabalho.

O YCSB deve fazer várias escolhas aleatórias quando está gerando cargas, tais como: qual operação realizar (inserção, atualização, leitura ou escaneamento), qual registro ler e quantos registros escanear. Estas decisões ficam a cargo de distribuições. O YCSB tem várias distribuições, sumarizadas na tabela 1 [27].

Neste trabalho, utilizou-se o *benchmark* YCSB (sessão 3. 1. 1), com 50 threads para gerar os dados. Os testes foram realizados com as cinco diferentes cargas de trabalho (A, B, C, D e E).

Para o teste de escalabilidade do Cassandra foram utilizados quatro computadores (processador athlon1Ghz, 2 GB de RAM,

rede megabit) e a carga escolhida foi a C, por ser uma carga de somente leitura.

Tabela 1. Cargas de trabalho do YCSB-Fonte: [27]

Carga de trabalho	Operações	Seleção de registros	Exemplo de aplicação
A – Muitas Atualizações	Leitura: 50% Atualização: 50%	Zipfian	Ações recentes de gravação e armazenamento de uma sessão de usuário
B – Muitas Leituras	Leitura: 95% Atualização: 5%	Zipfian	Marcação de foto; adicionar uma tag é uma atualização, mas a maioria das operações é para ler as *tags*.
C - Somente Leituras	Leitura: 100%	Zipfian	Cache de perfil de usuário, onde os perfis são construídos em outros lugares.
D - Leitura utilizando Ultima	Leitura: 95% Inserção: 5%	Ultima	Atualizações de status do usuário, as pessoas querem ler suas últimas atualizações.
E – Pequenos *Ranges*	Escaneamento: 95% Inserção 5%	Zipfian/Uniforme	Conversas por *threads*, onde cada varredura é para o *post* em uma determinada *thread*.

3.2 Métodos

Basicamente o trabalho se dividiu em três objetivos específicos diferentes: 1) Análise qualitativa do amadurecimento e consolidação dos softwares utilizados; 2) Testes de performance utilizando o YCSB e, 3) Teste de escalabilidade do software Cassandra.

A análise de amadurecimento e consolidação dos *softwares* avaliou qualitativamente os quesitos documentação *online*, *help* do *software*, suporte da comunidade virtual e artigos acadêmicos, comparando o Redis, Cassandra e MySQL.

Os testes de performance mediram o fluxo de dados e latências para 2000, 6000, 10000 e 14000 operações, utilizando o *benchmark* YCSB, com as cargas definidas na tabela 1. Os três SGBDs foram expostos ao teste.

O teste de escalabilidade foi realizado apenas com o SGBD Cassandra, pois o Redis utiliza replicação mestre/escravo, onde toda alteração no nó mestre é replicada aos nós escravos. O YCSB não atende bem esse cenário e por isso não foi testado. Já o MySQL, por ser relacional, não possui motivo para passar por um teste de escalabilidade horizontal. O teste de escalabilidade utiliza o mesmo princípio do teste de performance, no qual se mede a latência de leitura para 2000, 6000, 10000 e 14000 operações, utilizando o *benchmark* YCSB, com as cargas de trabalho definidas na tabela 1. No entanto, para cada nó adicionado, realiza-se a carga e suas operações e, após as operações, outro nó é adicionado ao *cluster* e o processo repete-se sucessivamente para cada nó.

4. RESULTADOS E DISCUSSÃO

4.1 Amadurecimento e consolidação dos softwares testados

Com relação à documentação online, foram avaliados pontos como a facilidade para encontrar tais documentos, a clareza dos mesmos e o desenvolvimento desta documentação.

O Redis apresenta em seu próprio site uma documentação muito boa sobre o *software*, com tutoriais de instalação, lista de todos os comandos, tutoriais para configuração e links para livros tanto para iniciantes, quanto para usuários experientes. O Cassandra conta com uma documentação *online* que não é tão completa e objetiva. Não é hospedada em seu site, mas sim no site da datastax[1]. A documentação do Cassandra vem sofrendo atualizações, inclusive algumas durante a escrita deste trabalho, que foram muito bem recebidas pela comunidade. O MySQL apresenta a melhor documentação *online* dentre os três bancos de dados estudados. Percebe-se que muitas pessoas contribuem com sua documentação. Os tópicos são mais bem organizados acarretando uma maior facilidade para encontrar a informação desejada. Essa documentação tão completa se deve ao tempo de mercado do *software* que é muito mais antigo que os outros dois apresentados neste trabalho.

Com relação ao sistema de ajuda do software (*help*), o Redis foi qualitativamente avaliado com o pior *help* entre os três SGBDs. Ainda faltam alguns elementos que os outros bancos de dados apresentam, como a apresentação de um exemplo demonstrando o comando requisitado e uma descrição geral deste. No geral é possível apenas sanar algumas dúvidas de sintaxe. O Cassandra apresenta um *help* considerado bom, melhor que sua documentação no site. Ele se aparenta muito com o *help* do MySQL, pois contém exemplos e explicações, ajudando muito os usuários inexperientes. O MySQL conta com o melhor *help* que, como já dito, possui explicações e exemplos, o que ajuda muito iniciantes e desenvolvedores.

No quesito de suporte da comunidade, foi avaliado o quanto as comunidades virtuais ajudam e o grau de comprometimento de seus participantes. O Redis possui a menor comunidade e talvez isso se deva ao fato dele ser chave-valor e ser simples. Mas é uma comunidade ativa, disposta a ajudar na solução dos problemas. Foi observado a falta de uma comunidade brasileira. No grupo de NoSQL do Facebook, não se encontra nenhuma discussão sobre o Redis. Pela importância do SGBD, espera-se que num futuro próximo surja uma comunidade nacional. O Cassandra, em conjunto com o MongoDB, são os SGBDs com a maior comunidade no ramo NoSQL. Como o Cassandra utiliza o modelo orientado a coluna, operações mais complexas podem ser realizadas com ele, o que gera mais dúvidas e discussões em

[1]http: //www. datastax. com/documentation/gettingstarted/index. html

fóruns. No Facebook existe uma comunidade brasileira. Observou-se que as comunidades do Cassandra são ativas e que, como o *software* está em pleno desenvolvimento, a comunidade tende sempre a amparar os desenvolvedores. Já o MySQL possui a maior comunidade, tanto em fóruns brasileiros, quanto em fóruns internacionais, grupos do Facebook e outras redes sociais. Os fatores que geram tal situação são dois: maturidade do MySQL e sua grande utilização. O MySQL começou a ser desenvolvido em 1994 enquanto bancos de dados NoSQL surgiram nos anos 2000.

No quesito de artigos acadêmicos, objetivou-se verificar a quantidade de artigos científicos publicados sobre os SGBDs testados. Os dados foram coletados do *Web of Science*,[2] um portal que permite o acesso a bases de dados de diversos periódicos. Para cada SGBD realizou-se a pesquisa utilizando o filtro: *Computer Science* e Nome do SGBD, compreendidos entre os anos de 2009 a 2013. A Figura 1 mostra o número de artigos referentes aos três SGBDs.

Verifica-se que, por ser tecnologias recentes, os bancos NoSQL testados ainda apresentam baixo número de artigos científicos na base consultada. Devido a sua maturidade e grande utilização, o MySQL apresenta um destaque maior em publicações periódicas quando comparado aos outros dois bancos de dados. Nessas consultas, o MySQL apresentou mais artigos relacionados a aplicações, enquanto o Cassandra apresentou mais artigos relacionados a estudos comparativos e testes de performance.

4.2 Testes de Performance

Nesta seção são apresentados os resultados de testes de performance realizados com os SGBDs Cassandra, MySQL e Redis. Os testes foram realizados medindo o fluxo de dados em relação à latência. O fluxo de dados está separado da latência, mas estão relacionados pelo número de operações, com o objetivo de demonstrar como os SGBDs se portam.

Foi utilizado o YCSB Client para a realização dos testes, com suas cargas padrões (Tabela 1). Os testes abaixo apresentados serão comparados ao de [27], que realizaram testes de performance com o MySQL e o Cassandra e teste de escalabilidade com o Cassandra.

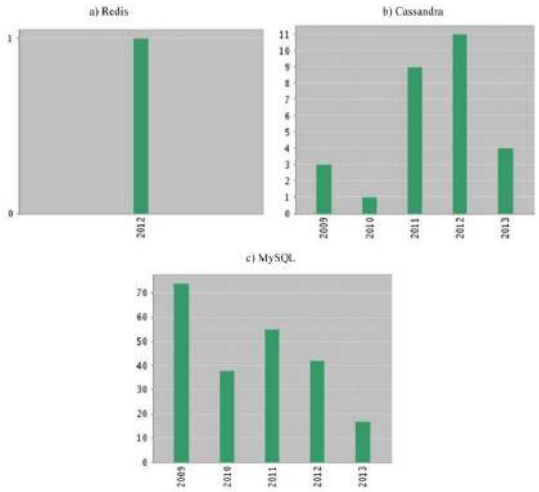

Figura 1: Número de artigos publicados por ano a) Redis, b) Cassandra, c) MySQL - Fonte: Web of Science

[2] : <http: //webofknowledge. com/>

4.2.1 *Carga A – Muitas Atualizações*

Os três bancos de dados foram testados com a carga A, que consiste de 50% de leitura e 50% de atualizações. Foram testados com 2000, 6000, 10000 e 14000 operações.

A Figura 2a demonstra o fluxo de dados em relação ao número de operações. Observa-se que o Redis apresenta o maior fluxo de dados, ou seja, ele é capaz de realizar um maior número de operações por segundo. De fato, em seus testes, [28] observou que durante a fase de carga (leitura), o Redis conseguiu alcançar 430 operações por segundo e de forma bastante estável em uma configuração single-threaded. Configurações multi-threaded rapidamente chegaram a um ponto de inflexão de mais de 9000 operações por segundo em cerca de 16 threads. Ainda segundo o autor, 28 threads parecia ser o ponto ideal entre cliente e servidor.

A latência mede o tempo de resposta do sistema. Assim, quanto maior a latência, pior o resultado. A Figura 2b apresenta a latência de atualizações em relação ao número de operações. O Redis e o Cassandra diferem numa média de 2ms. Os resultados obtidos com o Cassandra assemelham-se muito ao padrão apresentado por [27]. E os resultados obtidos com o Redis foram corroborados por [28].

A Figura 2c apresenta a latência de leitura em relação ao número de operações. Com relação às latências de atualização, o MySQL apresentou as maiores latências (pior resultado) e, para leitura, o Cassandra apresentou as maiores latências. Sendo que na leitura o MySQL diferiu do Redis em uma média de 3 ms. Neste teste percebe-se que o hardware influiu nos teste pois os resultados se portam como os de [27], mas com latências muitos maiores. Já o Redis, acompanhou os resultados obtidos por [28].

Considerando a máquina utilizada para teste não possui um hardware que se compara a de um servidor, os resultados para a carga A corroboram, de maneira geral, os testes de [27] para o MySQL e o Cassandra. O Redis apresenta a melhor performance, com as menores latências e o maior fluxo de dados. Com 14. 000 operações, ele triplica o número de operações do Cassandra. O Cassandra mantém um fluxo de dados maior que o MySQL e com uma boa latência de atualização. Porém o MySQL tem uma melhor latência de leitura, apesar de um menor fluxo de dados.

4.2.2 *Carga B – Muitas Leituras*

A carga B apresenta outro cenário para os testes, com 95% de carga de leitura e 5% de atualizações. A Figura 3a demonstra o fluxo de dados dos SGBDs. O Redis apresenta o melhor fluxo de dados, triplicando o número de operações, comparado ao MySQL, que se comporta muito melhor que o Cassandra em cenários de muitas leituras. Em 14. 000 operações quase que duplica o número de operações do Cassandra.

Com relação à latência de atualização, a Figura 3b mostra que o Cassandra e o Redis apresentam latências muito semelhantes variando em média 3ms. O MySQL apresenta as maiores latências de atualização e, quando comparado ao Cassandra chega a ser 15 vezes maior. Os dados corroboram os trabalhos de [27 e 28] quando se trata de atualizações. No entanto, nos testes de [27] o MySQL não apresenta valores tão altos, mas manteve-se melhor que o Cassandra. Como este é um teste de muitas leituras, o MySQL demonstra sua vantagem de leitura em relação ao Cassandra (Figura 3c).

No geral, em relação à carga de testes B, o Redis tem as melhores performances. O MySQL obteve desempenho melhor que o Cassandra que, em seu pico, apresenta latência 8 vezes e meia maior que a do MySQL. Esse teste evidencia a vantagem do MySQL para leitura quando comparado ao Cassandra. O MySQL

tem latências muito parecidas com a de [27]. O Cassandra com 2000 operações obteve 13 ms e na literatura de [27], 7 ms.

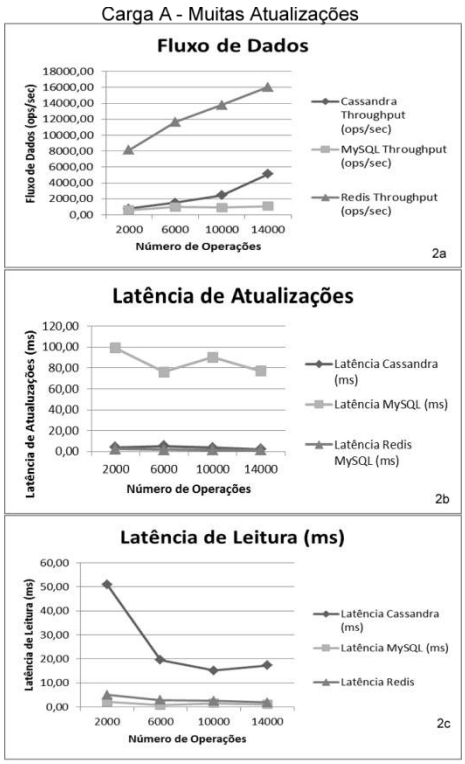

Figura 2: Resultado do teste de performance nos SGBDs MySQL, Redis e Cassandra para a carga A - muitas atualizações. a)Fluxo de dados por número de operações; b) Latência de atualização por número de operações; c) Latência de leitura por número de operações.

4.2.3 *Carga E – Pequenos Ranges*

Para a carga E foi utilizado o Redis e o Cassandra, pois o MySQL não suporta escaneamento de ranges. O fluxo de dados de ambos foi semelhante ao final do teste, porém no início o Redis apresentou uma diferença significativa no fluxo dos dados, como visto na Figura 4a. Com relação à latência dos escaneamentos ambos os SGBDs tendem a convergir para valores entre 100 e 120 ms (Figura 4b).

4.2.4 *Outras Cargas*

As cargas C (somente leitura) e D (leitura utilizando Ultima) apresentaram resultados muito semelhantes à carga B, por tratar simplesmente de mudança na porcentagem de leitura ou a ordem em que os dados são lidos. Uma ressalva vale ao MySQL, que na carga D obteve uma performance muito satisfatória.

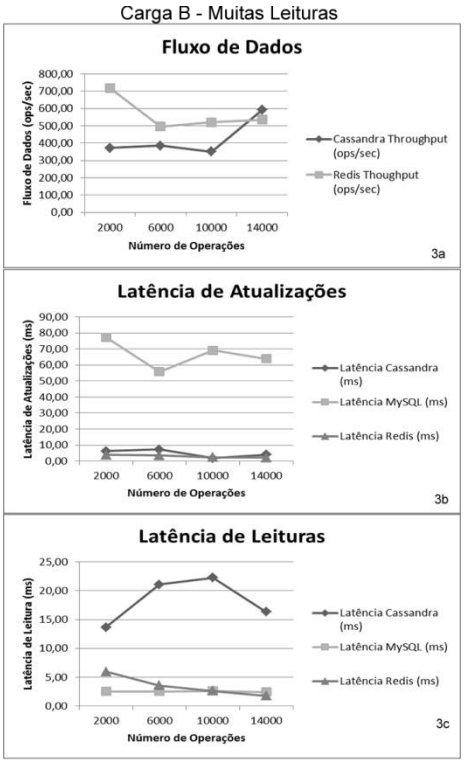

Figura 3 : Resultado do teste de performance nos SGBDs MySQL, Redis e Cassandra para a carga B - muitas leituras. a)Fluxo de dados por número de operações; b) Latência de atualização por número de operações; c) Latência de leitura por número de operações.

4.3 Teste de Escalabilidade

Considerando que sistemas NoSQL são construídos para escalar o número de nós, este teste teve por finalidade mensurar o desempenho de . O resultado é demonstrado na Figura 5.

Apesar dos testes apresentarem alta latência, o fato a observar é que o sistema manteve-se estável à medida que novos nós foram adicionados. Os resultados coincidem com os da literatura de [27], que realizou seus testes utilizando 12 computadores e ainda assim o sistema manteve-se estável, evidenciando a capacidade do Cassandra de escalar, manter-se estável e sem perda de performance.

Em relação aos testes de performance e de escalabilidade, observou-se que o Redis é um dos mais rápidos devido ao fato de ser chave-valor e operar na memória RAM. O Cassandra apresenta uma boa performance quando o quesito é atualização e observa-se que ele tem grande capacidade de escalabilidade. O MySQL tem vantagem na leitura devido a sua indexação por B-TREE, que fornece baixa latência para as leituras, proporcionando assim uma vantagem quando comparado ao Cassandra. Mesmo os testes não sendo realizados em máquinas que refletem as utilizadas em clusters, foi possível observar a diferença entre os três SGBDs.

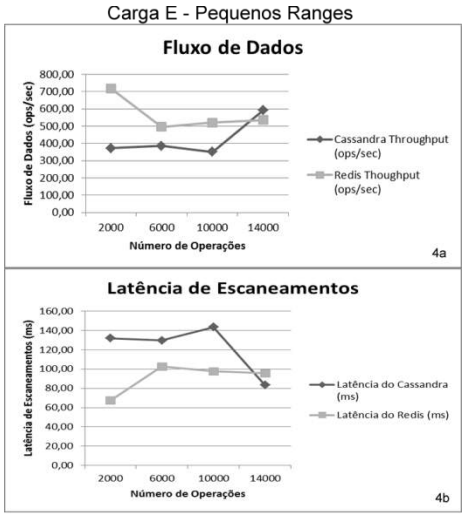

Figura 4 : Resultado do teste de performance nos SGBDsRedis e Cassandra para a carga E – pequenos Ranges. a) Fluxo de dados por número de operações; b) Latência de Escaneamento por número de operações.

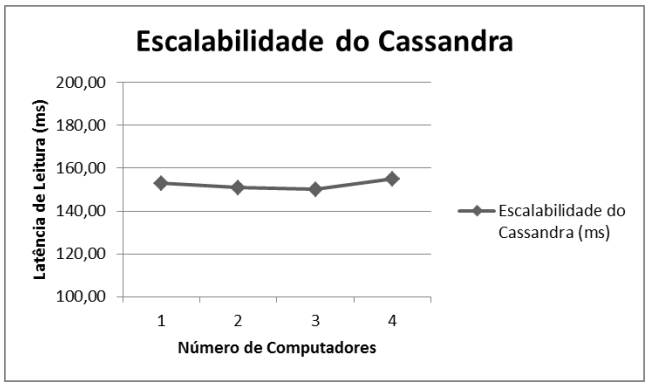

Figura 5 : Latência de leitura do SGBD Cassandra pelo número de nós.

5. CONSIDERAÇÕES FINAIS

Bancos de dados NoSQL oferecem uma alternativa aos bancos relacionais, para aplicações onde é necessário lidar com um grande volume de dados e, consequentemente, necessário escalar o servidor de banco com facilidade e eficiência. Essa nova tecnologia não substitui, mas complementa os bancos de dados relacionais, que ainda predominam tanto em pesquisas como no mercado. Porém, a cada dia, o NoSQL ganha mais espaço na comunidade, principalmente em grandes empresas e em aplicações da web 2. 0.

O estudo realizado nesse trabalho teve dois objetivos principais: verificar o amadurecimento e consolidação dos SGBDs NoSQL, e testar a performance e escalabilidade dos mesmos. No quesito amadurecimento e consolidação, ficou claro que, apesar de recente, a tecnologia vem se consolidando fortemente. Percebe-se que as pesquisas na área estão avançando e são guiadas, principalmente por grandes empresas, como Facebook e Twitter, entre outras, que necessitam dessa tecnologia. Comparando os SGBDs NoSQL Cassandra e Redis, mesmo tendo sido lançados com apenas um ano de diferença, verifica-se que o Cassandra apresenta maior consolidação, tanto no mercado, quanto na área acadêmica. O motivo talvez seja o próprio modelo de dados orientado à coluna, que permite uma gama maior de aplicações, comparado ao modelo de dados do Redis (chave-valor).

No quesito performance, o Redis mostrou-se superior, tanto no fluxo de dados, quanto na latência. O motivo de tão bom desempenho deve-se ao seu modelo de dados e a sua estrutura de funcionamento. O redis utiliza tabela hash e tenta manter o conjunto de dados em memória RAM. Essa combinação o fez ser muito superior ao Cassandra na maioria dos testes. Um resultado significativo e que merece ser destacado foi o desempenho do MySQL nas cargas de leitura, que mostrou que o algoritmo de indexação desse software é altamente competitivo e eficiente.

Já no quesito escalabilidade, esse trabalho teve como objetivo por à prova a facilidade de escalonamento horizontal dos bancos NoSQL. Ficou evidente essa propriedade no Cassandra. A facilidade de inserir e remover nós do cluster é clara. O próprio servidor se encarrega de distribuir os dados e as tarefas. Sendo a escalabilidade o motivo principal do aparecimento dos bancos NoSQL, conclui-se que o Cassandra cumpre seu papel e representa uma ótima solução para escalar aplicações Big Data.

É importante esclarecer que os bancos NoSQL e os relacionais têm seus próprios nichos de mercado. A alta disponibilidade, escalabilidade, flexibilidade do esquema e alta performance e gerenciamento de dados dos bancos NoSQL, trazem a desvantagem de nem sempre ser possível garantir a consistência dos dados. Assim, torna-se evidente a necessidade do conhecimento e entendimento de ambas tecnologias para decidir sobre qual adotar em cada situação específica. Para auxiliar nessa tomada de decisão, [29] pesquisaram e chegaram a 22 critérios que devem ser observados pelas organizações em seus processos de seleção de SGBDs NoSQL.

6. REFERÊNCIAS

[1] Elmasri, R; Navathe, S. B. *Sistemas de banco de dados*. 6 ed. São Paulo: Pearson 2011.

[2] Korth, H. F. ; Silberschatz, A. ; Sudarshan, S. *Sistema de banco de dados*. 5 ed. Rio de Janeiro: Elsevier, 2006. 781 p.

[3] Vieira, M. ; Figueiredo, J. Bancos de Dados NoSQL: Conceitos, Ferramentas, Linguagens e Estudos de Casos no Contexto de Big Data. In *Anais do Simpósio Brasileiro de Bancos de Dados* (São Paulo/SP - Brasil, 15-18/10/2012). Disponível em: http://data.ime.usp.br/sbbd2012/artigos/pdfs/sbbd_min_01.pdf. Acesso em: 31 Jan. 2015.

[4] DB-Engines. Ranking. Disponível em: http://db-engines.com/en/ranking. Acesso em: 3 Set. 2013.

[5] Fowler, M. *NoSQL*. Disponível em: http://martinfowler.com/nosql.html Acessado em: 31 Jan. 2015.

[6] D'Andrea, E. Big Data. *Revistainformationweek*, 2010. Disponível em: http: //www.pwc.com.br/pt/sala-de-imprensa/assets/artigo-big-data.pdf. Acessoem: 31 Jan. 2015.

[7] Singh, S. ; Singh, N. Big Data Analytics. In *Anais de International Conferenceon Communication, Information & Computing Technology* (Mumbai, India, Oct. 19-20/2012) - ICCICT. DOI : http://dx doi.org/10.1109/ICCICT.2012.6398180

[8] Han, J; Haihong, E; Le, G; Du, J. Surveyon NoSQL database. In Anais de *Pervasive computing and Applications*(Port Elizabeth – África do Sul, 26-28 Out. 2011), p. 363-366, 2011. DOI: http://dx.doi.org/10.1109/ICPCA.2011.6106531.

[9] Brito, R. W. *Bancos de Dados NoSQL x SGBDs Relacionais: Análise Comparativa*. Universidade de Fortaleza, 2010. Disponível em: http://www.infobrasil.inf.br/userfiles/27-05-S4-1-68840-Bancos de Dados NoSQL.pdf. Acesso em: 31 Jan. 2015.

[10] Leavitt, N. Will NoSQL Databases Live Up to Their Promise? *Computer*, v. 43, n. 2, p. 12–14, Fev. 2010. DOI: http://dx.doi.org/10.1109/MC.2010.58

[11] Fowler, M; sadalage, P. *NoSQL Databases Polyglot Persistence SQL has Ruled for two decades*. 2012. Disponível em: http://martinfowler.com/articles/nosql-intro-original.pdf. Acesso em: 31 Jan. 2015.

[12] Pritchett, D. Base: An acid alternative. *Queue – Object-Relational Mapping*. v. 6, n. 3, p48-55. Maio/Jun 2008. DOI: http://dx.doi.org/10.1145/1394127.1394128

[13] Silva, C. A. R. F. O. *Data Modeling with NoSQL : How, When and Why* (Dissertação). Universidade do Porto. 2011. Disponível em : http://repositorio-aberto.up.pt/handle/10216/61586. Acesso em: 31 Jan. 2015.

[14] Brewer, E. A. Towards Robust Distributed Systems, In *Anais de 19th Ann. ACM Symp. Principles of Distributed Computing* (PODC 00), ACM, 2000, pp. 7-10. DOI : http://dx.doi.org/10.1145/343477.343502

[15] Brewer, E. A. CAP Twelve Years Later: How the "Rules" Have Changed. *Computer*, v. 45, n. 2, p. 23–29, Fev. 2012. DOI: http://dx.doi.org/10.1109/MC.2012.37

[16] Abadi, D. ; Madden, S. ; Ferreira, M. Integrating compression and execution in column-oriented database systems. In Anais do ACM SIGMOD international conference on Management of data (New York, New York, USA. 2006). DOI : http://dx.doi.org/10.1145/1142473.1142548.

[17] Abadi, D. ; Boncz, P. ; Harizopoulos, S. Column-oriented database systems. In Anais VLDB Endowment. v. 2, n. 2, Ago. 2009. DOI : http://dx.doi.org/10.14778/1687553.1687625.

[18] Dwivedi, A. Performance Analysis of Column Oriented Database Vs Row Oriented Database. *International Journal of Computer Applications*, v. 50, n. 14, p. 31–34, 2012. DOI : http://dx.doi.org/10.5120/7841-1050

[19] Pokorny, J. NoSQL databases: a step to database scalability in web environment. *International Journal of Web Information Systems*, v. 9, n. 1, p. 69–82, 2013 DOI : http://dx.doi.org/10.1108/17440081311316398.

[20] Bonnet, L. ; laurent, A. ; sala, M. ; laurent, B. ; sicard, N. Reduce, You Say: What NoSQL Can Do for Data Aggregation and BI in Large Repositories. In *Anais do 22 ^{o}International Workshop on Databaseand Expert Systems Applications* (Toulouse, França. 29 Ago – 2 Set 2011) p. 483–488, 2011.

[21] Indrawan-Santiago, M. Database Research: Are Weat a Crossroad? Reflection on NoSQL. Anais do *15^{o}InternationalConferenceon Network-Based Information Systems (NBiS)* (Melbourne, VIC, 26-28 Set. 2012). DOI : http://dx.doi.org/10.1109/NBiS.2012.95

[22] Lee, K. K. -Y. ; tang, W. -C. ; choi, K. -S. Alternatives to relational database: comparison of NoSQL and XML approaches for clinical data storage**.** *Computer methods and programs in biomedicine*, v. 110, n. 1, p. 99–109, 2013. DOI : http://dx.doi.org/10.1016/j.cmpb.2012.10.018

[23] Paksula, M. Persisting Objects in Redis Key-Value Database. University of Helsinki, Department of Computer Science, 2010.

[24] Cattell, R. Scalable SQL andNoSQL data stores. *ACM SIGMOD Record*. v. 39, n. 4, p. 12-27. Dez 2011. DOI : http://dx.doi.org/10.1145/1978915.1978919

[25] Lakshman, A. ; malik, P. Cassandra: a decentralized structured storage system*. ACM SIGOPS Operating Systems Review*. v. 44, n. 2, p. 35-40. Abril 2010. DOI : http://dx.doi.org/10.1145/1773912.1773922

[26] Fior, A. G. F. ; Meira, J. A. ; Almeida, E. C. ; Coelho, R. G. ; Del Fabro, M. D. ;Traon, Y. L. Underpressure benchmark: a large-scale availability benchmark for distributed databases. *Journal of Information and Data Management*. v. 4, n. 3, p. 266-278. Out. 2013. Disponível em : https://seer.lcc.ufmg.br/index.php/jidm/article/view/249/199. Acesso em: 31 Jan. 2015

[27] Cooper, B. ; Silberstein, A. ; Tam E. ; Ramakrishnan, R. ; Sears, R. Benchmarking cloud serving systems with YCSB. In Anais do 1º ACM symposium on Cloud computing (Indianápolis, In, 10-11/06/2010 p. 143-154). DOI : http://dx.Doi.org/10.1145/1807128.1807152

[28]Lehmann , R. Redis in the Yahoo! Cloud ServingBenchmark. 2011. [online] Disponível em : http://robertlehmann.de/img/redis.pdf. Acesso em: 31 Jan. 2015.

[29] Alexandre Morais de Souza, Edmir P. V. Prado, Violeta Sun, Marcelo Fantinato. Critérios para Seleção de SGBD NoSQL: o Ponto de Vista de Especialistas com base na Literatura. In Anais doX Simpósio Brasileiro de Sistemas de Informação (Londrina –PR, Brasil. 27 a 30/05/2014