NOSQL na Web 2.0: Um Estudo Comparativo de Bancos Não-Relacionais para Armazenamento de Dados na Web 2.0

Mauricio De Diana[1], **Marco Aurélio Gerosa**[1]

[1]Departmento de Ciência da Computação – Universidade de São Paulo (USP)
São Paulo – SP – Brazil

{mdediana,gerosa}@ime.usp.br

Abstract. *With Web 2.0, the amount of data created, stored and processed has reached a scale never seen before. Web 2.0 organizations have been implementing and using solutions beyond relational databases to tackle data on this magnitude. These non-relational databases were named NOSQL. To better understand their use cases, it is necessary to categorize, classify and compare them by both the data models they provide, and their scalability and performance attributes.*
This article presents a taxonomy of these new technologies, their main differences from relational databases, and the trade-offs between scalability and other system quality attributes they make.

Resumo. *A quantidade de dados gerados, armazenados e processados atingiu escalas inéditas com a Web 2.0. Soluções além dos bancos de dados relacionais estão sendo implementadas e utilizadas por organizações da Web 2.0 para lidar com dados desse volume. Ao conjunto desses bancos de dados não-relacionais se deu o nome de NOSQL. Para entender seus casos de uso, é necessário categorizá-los, classificá-los e compará-los com relação a seus modelos de dados e seus atributos de escalabilidade e performance.*
Esse artigo apresenta uma taxonomia dessas novas tecnologias, suas principais diferenças com relação aos bancos de dados relacionais e como lidam com o compromisso entre escalabilidade e outros atributos de qualidade de sistemas.

1. Introdução

O mundo nunca lidou com volumes de dados tão grandes, em parte graças a Web 2.0. No artigo em que define Web 2.0, Tim O'Reilly fala sobre a necessidade de se aproveitar a Inteligência Coletiva e sobre dados como diferencial competitivo [O'Reilly 2005]. Não por acaso muitas inovações na área de gerenciamento de dados vieram de algumas empresas pioneiras da Web 2.0, que encontrando limites nas técnicas e ferramentas disponíveis naquele momento, criaram sua próprias soluções, em sua maioria não-relacionais. Boa parte da motivação por trás dessas soluções estava ligada a novos requisitos de escalabilidade e disponibilidade, que fizeram com que outros requisitos tidos como indiscutíveis fossem revistos, como a consistência dos dados, por exemplo [Vogels 2009, Pritchett 2008].

Seguindo essa onda de inovações no gerenciamento de dados na web vieram organizações menores e a comunidade de software livre e de código aberto em geral, que inspiradas naquelas primeiras ideias, criaram diversas soluções de bancos de dados não-relacionais, seguindo diferentes paradigmas. Apesar de bancos de dados não-relacionais

não serem novidade, inclusive existindo antes dos bancos de dados relacionais, a esse conjunto específico de soluções surgidas nessa segunda onda na Web 2.0 deu-se o nome NOSQL. Mesmo ainda estando cercadas de polêmicas e discussões [Stonebraker 2009, Leavitt 2010, Stonebraker 2010], tanto academia quanto mercado reconhecem a importância e a necessidade de estudá-las [Agrawal et al. 2009, Release 2009, Radar 2010].

Como essas tecnologias são muito recentes, existem poucas recomendações indicando em que contexto usar determinado paradigma. Em especial, apesar de haver casos em que bancos de dados NOSQL ajudaram organizações a escalar seus sistemas, existem poucos estudos comparativos que indiquem os cenários em que se aplicam e quais são seus limites de performance e escalabilidade. Esse trabalho tem por objetivo investigar e testar os bancos de dados NOSQL do ponto de vista de escalabilidade e performance, e gerar recomendações sobre seu uso.

Esse artigo introduz algumas dessas novas tecnologias e apresenta suas diferenças com relação aos bancos de dados relacionais. A seção 2 apresenta os bancos de dados NOSQL e analisa o contexto da Web 2.0 que provavelmente motivou o surgimento e utilização em larga escala destas tecnologias. A seção 3 fala sobre o conceito de consistência em momento indeterminado, a seção 4 apresenta as principais categorias de bancos de dados NOSQL e a seção 5 apresenta os trabalhos futuros que pretendemos realizar nessa área.

2. Bancos de Dados NOSQL

Os bancos de dados NOSQL surgiram como uma solução para a questão da escalabilidade no armazenamento e processamento de grandes volumes de dados na Web 2.0. No início, grandes empresas enfrentando esse tipo de problema criaram suas próprias soluções, e publicaram alguns artigos científicos descrevendo diversas soluções ligadas ao gerenciamento de dados distribuído em larga escala, mas sem usar ainda o nome NOSQL [Chang et al. 2006, DeCandia et al. 2007, Cooper et al. 2008]. O nome só surgiu alguns anos depois, em 2009, quando algumas novas empresas da Web 2.0 e a comunidade de software livre e código aberto começaram a desenvolver novas opções de bancos de dados, inspiradas nas ideias que apareceram naqueles artigos.

Assim como o termo não-relacional, o termo NOSQL não ajuda a definir o que esses bancos são de fato. Além do problema da falta de precisão, esse termo também tem contribuído para uma grande confusão em torno dessa categoria de bancos de dados, já que a princípio a linguagem SQL não é sinônimo de bancos de dados relacionais, nem representa as limitações desses bancos de dados. Devido a isso, o termo NOSQL tem sido usado com o significado de "Não apenas SQL" [1] numa tentativa da comunidade de reconhecer a utilidade dos modelos tradicionais e não divergir as discussões. No fim, NOSQL não define precisamente esses bancos de dados, mas no geral cada um deles apresenta a maioria das seguintes características: não-relacional, distribuído, de código aberto e escalável horizontalmente, ausência de esquema ou esquema flexível, suporte à replicação nativo e acesso via APIs simples [NOSQL 2010]. Entre os principais fatores que favoreceram seu surgimento estão a natureza dos dados da web, a importância de se atingir altos graus de paralelismo no processamento de dos grandes volumes de dados e a distribuição de sistemas em escala global.

[1] "Not only SQL", em inglês.

Natureza dos dados da web A web é composta por uma grande quantidade de dados semiestruturados e crús, como as páginas web (cuja estrutura descrita no documento HTML expressa muito pouco sobre o significado do conteúdo do documento) e conteúdo multimídia (imagens, sons e vídeos). Ao reconhecer a natureza particular dos dados é possível criar soluções otimizadas para eles, ao invés de se tentar estruturá-los, como proposto por alguns autores [Atzeni et al. 1997]. A constatação de que os dados na web não são estruturados é um dos fatores que favoreceram o surgimento de tecnologias de gerenciamento de dados diferentes das tradicionais.

Importância do paralelismo para o processamento de grandes volumes de dados O uso de hardware comum e barato para operação dos sistemas é outro fator que levou ao surgimento de soluções alternativas para o gerenciamento de dados. Para se atingir performance razoável no processamento de grandes volumes de dados é necessário pensar nos sistemas de forma a se atingir alto grau de paralelismo. Quando esse paralelismo é factível, o uso de muitos processadores baratos não só oferece melhor performance, mas também é uma solução economicamente mais interessante que o uso de menos processadores mais poderosos e mais caros [Barroso et al. 2003]. Dessa forma, é possível escalar o sistema horizontalmente apenas adicionando mais hardware. Além da questão do paralelismo, escalabilidade horizontal é importante para que o crescimento de uma organização não fique limitado a capacidade de fornecedores criarem hardware mais poderoso [Pritchett 2008, Abbott and Fischer 2009].

Distribuição dos sistema em escala global Para atender seus usuários de forma eficiente, algumas organizações utilizam diversos datacenters, muitas vezes localizados em países diferentes. Isso introduz uma série de preocupações sobre disponibilidade e performance [Vogels 2009] que devem ser levados em consideração na construção dos sistemas. A distribuição dos sistemas globalmente e o uso de hardware comum e barato impõem um novo requisito sobre o software sendo implementado: ele deve ser robusto o suficiente para tolerar constantes e imprevisíveis falhas de hardware e de infraestrutura (como redes de terceiros, por exemplo). Isso levou à construção de muita infraestrutura por parte de grandes empresas para dar suporte aos pesados requisitos de suas aplicações, como o GFS, por exemplo [Ghemawat et al. 2003]).

3. Consistência em momento indeterminado

Uma propriedade que muitos bancos de dados NOSQL têm em comum é a consistência em momento indeterminado (*eventual consistency*). Essa propriedade é fundamental para que algumas das soluções consigam atingir níveis maiores de escalabilidade.

As propriedades de transações ACID (Atômica, Consistente, Isolada e Durável) tornam o trabalho do desenvolvedor de aplicações muito mais simples. Mas apesar de muito desejáveis, essas propriedades criam dificuldades ao se distribuir o banco de dados. Quando um banco de dados relacional cresce além da capacidade de um único nó é preciso se optar por escalabilidade vertical ou horizontal. Escalabilidade vertical não é uma opção para sistemas que lidam com grandes volumes de dados. Assim, a opção é escalar horizontalmente, e nesse caso há duas formas de se particionar os dados [Pritchett 2008]. A primeira é o particionamento funcional, que consiste em distribuir as tabelas pelos nós

de acordo com as funcionalidades do sistema que elas atendem, por exemplo, um nó pode conter as tabelas relacionadas a usuários e outro as relacionadas a cobrança. O segundo tipo, chamado *sharding*, ocorre quando uma tabela está dividida em mais de um nó. Quando essa técnica é utilizada, os dados são espalhados por nós de acordo com um critério arbitrário qualquer, como a inicial do nome do usuário em uma tabela de usuários, por exemplo. Ao se aplicar *sharding* em um banco de dados, mesmo com a falha de um nó o sistema continua funcionando para todas as operações que não dependam dos dados contidos naquele nó. A desvantagem dessa estratégia é que o banco de dados perde parte da sua capacidade de lidar com restrições dos dados, além de não serem mais capazes de realizar JOINs transparentemente. Com isso, os bancos de dados relacionais deixam de oferecer algumas de suas principais funcionalidades para o desenvolvedor de aplicações, que passa a ter que fazer esses tratamentos no nível da aplicação. Ambas as técnicas foram e ainda são muito utilizadas por sistemas da Web 2.0.

O teorema CAP fala sobre esse tipo de situação [Brewer 2000, Gilbert and Lynch 2002]. As três letras de CAP se referem à consistência, disponibilidade e tolerância a partição [2]. Consistência nesse contexto não tem exatamente o mesmo significado da consistência de transações de bancos de dados, mas sim diz respeito à ordem de execução de requisições, e significa que uma leitura de um item após uma escrita desse item deve retornar o novo valor. Disponibilidade é a propriedade de um sistema responder a todas as requisições que chegam a um nó funcionando. Tolerância à partição é a propriedade de um sistema continuar funcionando mesmo quando um problema ocorre na rede dividindo o sistema em duas ou mais partições, o que faz com que nós de uma partição não consigam se comunicar com as outras. Em sistemas tolerantes à partição clientes acessando uma partição conseguem ser atendidos normalmente. O teorema CAP diz que, em um sistema distribuído, só é possível garantir duas dessas três propriedades em um dado instante.

Partições de rede são raras mas ocorrem de tempos em tempos em sistemas largamente distribuídos (sistemas distribuídos por vários datacenters, por exemplo) [Vogels 2009]. A partir dessa constatação, para essa categoria de sistemas é necessário escolher entre disponibilidade e consistência. Dada a natureza das aplicações Web 2.0 elas costumam optar por disponibilidade quando for possível tolerar alguma inconsistência temporária. Para descrever essas situações, Pritchett fala sobre BASE [Pritchett 2008], em oposição a ACID [3]. BASE significa basicamente disponível, estado leve e consistente em momento indeterminado [4], mas o termo não descreve propriedades de fato como as definidas por ACID. O termo apenas indica que deve-se planejar um sistema de forma a tolerar inconsistências temporárias quando se quer priorizar disponibilidade.

4. Taxonomia de Bancos de Dados NOSQL

Essa seção descreve os tipos mais comuns de bancos de dados NOSQL: bancos de dados orientados a documentos, armazéns de chave-valor, bancos de dados de famílias de colunas e bancos de dados de grafos. O principal objetivo é apresentar suas diferenças básicas com relação aos bancos de dados tradicionais, seu contexto de uso na Web 2.0 e suas principais implementações.

[2] *Consistency*, *availability* e *partition tolerance*, em inglês.

[3] Base versus ácido.

[4] *Basically available*, *soft state* e *eventually consistent*, em inglês

4.1. Bancos de dados orientados a documentos

Os documentos de bancos de dados orientados a documentos são 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.

É comum os bancos de dados orientados a documento usarem algum formato de armazenamento que suporte que uma estrutura seja embutida em outra (ou seja, um documento embutido em outro), como o JSON. Embutir um documento em outro muitas vezes leva à duplicação de dados. Duplicação de dados, por sua vez, pode criar problemas de consistência no banco de dados, causados por anomalias de atualização e deleção, por exemplo. A normalização em bancos de dados relacionais visa evitar esses problemas. Mas considerando que se perde parte das garantias de integridade ao se fazer *sharding* em um banco de dados relacional, a duplicação de dados em bancos de dados orientados a documentos não é um impeditivo. Além disso, a duplicação de dados facilita a distribuição do sistema, já que a quantidade de nós a serem consultados em uma busca envolvendo várias entidades relacionadas é menor caso elas estejam próximas – no caso em que todas estão embutidas umas nas outras, a consulta ocorre em um único nó. Esse fato aponta para uma característica presente não só nos bancos de dados orientados a documentos, mas em bancos de dados NOSQL em geral, que é a criação de modelos de dados que favoreçam o seu uso para leitura.

Bancos de dados populares nessa categoria são o MongoDB [5] e o CouchDB [6].

4.2. Armazéns chave-valor

Sistemas distribuídos nessa categoria, também conhecidos como tabelas de hash distribuídas, armazenam objetos indexados por chaves, e possibilitam a busca por esses objetos a partir de suas chaves. Num primeiro momento, o estudo e uso desse tipo de sistema estiveram bastante ligados ao problema da localização de objetos em redes peer-to-peer. O Dynamo é uma implementação da Amazon dessa categoria de banco de dados e foi a primeira descrição dessa categoria no contexto da Web 2.0 [DeCandia et al. 2007].

Alguns bancos de dados nessa categoria são RIAK [7], Redis [8] e MemcacheDB [9].

4.3. Bancos de dados de famílias de colunas

Bancos relacionais normalmente guardam os registros das tabelas contiguamente no disco. Por exemplo, caso se queira guardar id, nome e endereço de usuários em um sistema de cadastro, os registros seriam: *Id1, Nome1, Endereço1; Id2, Nome2, Endereço2*. Essa estrutura torna a escrita muito rápida, pois todos os dados de um registro são colocados no disco com uma única escrita no banco. Essa estrutura também é eficiente caso se queira ler registros inteiros. Mas para situações onde se quer ler algumas poucas colunas de muitos registros, essa estrutura é pouco eficiente, pois muitos blocos do disco

[5] `http://www.mongodb.org/`
[6] `http://couchdb.apache.org/`
[7] `http://wiki.basho.com/display/RIAK/`
[8] `http://code.google.com/p/redis/`
[9] `http://memcachedb.org/`

terão que ser lidos. Para esses casos onde se quer otimizar a leitura de dados estruturados, bancos de dados de famílias de colunas são mais interessantes, pois eles guardam os dados contiguamente por coluna [Stonebraker et al. 2005]. O exemplo anterior em um banco de dados dessa categoria ficaria: *Id1, Id2; Nome1, Nome2; Endereço1, Endereço2*. Por esse exemplo é possível perceber a desvantagem de um banco de dados de famílias de colunas: a escrita de um novo registro é bem mais custosa do que em um banco de dados tradicional. Assim, num primeiro momento, os bancos tradicionais são mais adequados a processamento de transações online (OLTP) enquanto os bancos de dados de famílias de colunas são mais interessantes para processamento analítico online (OLAP). O Bigtable é uma implementação da Google dessa categoria de bancos de dados [Chang et al. 2006].

Os principais bancos de dados nessa categoria são Cassandra [10], HBase [11] e Hypertable [12], implementação baseada no artigo sobre o Bigtable.

4.4. Bancos de dados de grafos

Diferentemente de outros tipos de bancos de dados NOSQL, esse está diretamente relacionado a um modelo de dados estabelecido, o modelo de grafos. A ideia desse modelo é representar os dados e / ou o esquema dos dados como grafos dirigidos, ou como estruturas que generalizem a noção de grafos [Angles and Gutierrez 2008]. Operações sobre os dados são transformações sobre o grafo, e fazem uso de conceitos de grafos, como caminhos, vizinhos e sub-grafos. Esse modelo também dá suporte ao uso de restrições sobre os dados, como restrições de identidade e de integridade referencial, por exemplo. O modelo de grafos é mais interessante que outros quando "informações sobre a interconectividade ou a topologia dos dados é mais importante, ou tão importante quanto, os dados propriamente ditos" [Angles and Gutierrez 2008].

Exemplos de bancos de dados nessa categoria são o Neo4j [13] e InfoGrid [14].

5. Trabalhos Futuros

Ainda existem poucas recomendações sobre quando usar (ou não usar) bancos de dados NOSQL. Muito do seu uso atual na Web 2.0 está ligado a cenários onde escalabilidade é a questão principal. Mas apesar de várias empresas apresentarem relatos de sucesso na migração de bancos de dados relacionais para NOSQL, e alegarem terem resolvido seus problemas de escalabilidade, ainda falta um melhor entendimento do contexto e dos limites desses bancos de dados.

Dessa forma, pretendemos executar experimentos que ajudem a comparar esses bancos de dados entre si e com bancos de dados tradicionais em diferentes contextos. Para tal, pretendemos criar diferentes cenários típicos de aplicações da Web 2.0 e testar as diversas categorias de bancos de dados dentro de cada um deles. Alguns exemplos desses cenários são o grafo de uma rede social, a execução de um algoritmo de recomendação em uma base de dados de produtos e a busca por informações em uma base de dados formada por posts e comentários em blogs.

[10] `http://cassandra.apache.org/`
[11] `http://hbase.apache.org/`
[12] `http://hypertable.org/`
[13] `http://neo4j.org/`
[14] `http://infogrid.org/`

Os testes devem gerar informações sobre o comportamento dos bancos de dados em diversas escalas. Assim, serão criados cenários de testes de diferentes escalas, variando a quantidade de nós para colher métricas. Além disso, pretendemos testar distribuições por diferentes localizações geográficas de nós para simular escala global. Existem algumas plataformas acadêmicas para testes como Open Cirrus [15] e PlanetLab [16] que permitem esse tipo de experimento, incluindo o uso de múltiplos datacenters.

Apesar de objetivo principal desses experimentos ser entender o comportamento de cada categoria com relação a escalabilidade e performance, a adequação do modelo de dados à situação em mãos também deve ser considerada.

6. Conclusões

Até pouco tempo a preocupação com armazenamento e processamento de grandes volumes de dados era exclusividade de grandes organizações. Mas o cenário mudou, em especial na Web 2.0, onde o tratamento de dados é diferencial competitivo. Nesse cenário, as tecnologias tradicionais para gerenciamento de dados passaram a apresentar limitações, particularmente no que se refere à escalabilidade, e novas opções surgiram.

Os bancos de dados NOSQL são muito recentes. Apesar de ainda estarem cercados por confusões e polêmicas, em especial devido ao termo NOSQL não descrever esses bancos de dados de forma precisa, há indícios de que seu futuro é promissor. Mas além de promissor, ele é incerto. Recentemente, além do surgimento de várias novas opções, já começamos a ver o desaparecimento de algumas soluções. Além disso, muitos desses bancos de dados estão sendo testados em ambientes reais de produção agora, e ainda não há uma conclusão final sobre suas características, em especial no que diz respeito a escalabilidade e performance. Mas no momento vemos negócios e operações de escala global construídos sobre essas tecnologias: Cassandra é usado no Facebook, Twitter e Digg, e o MongoDB está em uso no Foursquare e SourceForge, para citar dois exemplos. Esse fato é um indicativo de que essas tecnologias provavelmente não serão descartadas facilmente como uma simples moda.

Neste artigo foi apresentada uma taxonomia das soluções NOSQL, mapeando algumas das tecnologias encontradas e seus usos na Web 2.0. Espera-se com este estudo estruturar o conhecimento sobre este tipo de aplicações e abrir caminho para novas pesquisas nesta área.

Referências

Abbott, M. and Fischer, M. T. (2009). *The Art of Scalability: Scalable Web Architecture, Processes, and Organizations for the Modern Enterprise*. Addison-Wesley Professional, 1 edition.

Agrawal, R., Ailamaki, A., Bernstein, P. A., Eric A. Brewer, Carey, M. J., Chaudhuri, S., Doan, A., Florescu, D., Franklin, M. J., Garcia-Molina, H., Gehrke, J., Gruenwald, L., Haas, L. M., Halevy, A. Y., Hellerstein, J. M., Ioannidis, Y. E., Korth, H. F., Kossmann, D., Madden, S., Magoulas, R., Ooi, B. C., O'Reilly, T., Ramakrishnan, R., Sarawagi, S., Stonebraker, M., Szalay, A. S., and Weikum, G. (2009). The Claremont Report on Database Research. *Communications of the ACM*, 52(6):56.

[15] `http://opencirrus.org/`

[16] `http://www.planet-lab.org/`

Angles, R. and Gutierrez, C. (2008). Survey of Graph Database Models. *ACM Computing Surveys*, 40(1):1–39.

Atzeni, P., Mecca, G., and Merialdo, P. (1997). Semistructured and Structured Data in the Web: Going Back and Forth. *ACM SIGMOD Record*, 26(4):16–23.

Barroso, L. A., Dean, J., and Hölzle, U. (2003). Web Search for a Planet: The Google Cluster Architecture. *IEEE Micro*, 23(2):22–28.

Brewer, E. A. (2000). Towards Robust Distributed Systems. *Annual ACM Symposium on Principles of Distributed Computing*.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W., Wallach, D., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. (2006). Bigtable: A Distributed Storage System for Structured Data. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation (OSDI'06)*, volume 26, pages 1–26.

Cooper, B., Ramakrishnan, R., Srivastava, U., Silberstein, A., Bohannon, P., Jacobsen, H.-a., Puz, N., Weaver, D., and Yerneni, R. (2008). PNUTS: Yahoo!'s Hosted Data Serving Platform. In *Proceedings of the VLDB Endowment*, volume 1, pages 1277–1288. VLDB Endowment.

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., and Vogels, W. (2007). Dynamo: Amazon's Highly Available Key-value Store. *ACM SIGOPS Operating Systems Review*, 41(6):220.

Ghemawat, S., Gobioff, H., and Leung, S. (2003). The Google File System. *ACM SIGOPS Operating Systems Review*, 37(5):43.

Gilbert, S. and Lynch, N. (2002). Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Services. *ACM SIGACT News*, 33(2):51.

Leavitt, N. (2010). Will NoSQL Databases Live Up to Their Promise? *Computer*, 43(2):12–14.

NOSQL (2010). NOSQL Databases. http://nosql-database.org/.

O'Reilly, T. (2005). What Is Web 2.0. http://oreilly.com/lpt/a/6228.

Pritchett, D. (2008). BASE: An Acid Alternative. *Queue*, 6(3):48–55.

Radar (2010). ThoughtWorks Technology Radar. (April):8.

Release (2009). Release 2.0. (11).

Stonebraker, M. (2009). The "NoSQL" Discussion has Nothing to Do With SQL. http://cacm.acm.org/blogs/blog-cacm/50678-the-nosql-discussion-has-nothing-to-do-with-sql/fulltext.

Stonebraker, M. (2010). SQL databases v. NoSQL databases. *Communications of the ACM*, 53(4):10.

Stonebraker, M., Abadi, D., Batkin, A., Chen, X., Cherniack, M., Ferreira, M., Lau, E., Lin, A., Madden, S., O'Neil, E., and Others (2005). C-store: a Column-oriented DBMS. In *Proceedings of the 31st International Conference on Very Large Data Bases*, page 564. VLDB Endowment.

Vogels, W. (2009). Eventually Consistent. *Communications of the ACM*, 52(1):40.