GeoNoSQL: Banco de dados geoespacial em NoSQL

Luís E. O. Lizardo[1], **Mirella M. Moro**[1], **Clodoveu A. Davis Jr.**[1]

[1]Departamento de Ciência da Computação – Universidade Federal de Minas Gerais
Caixa Postal 702 – 30.123-970 – Belo Horizonte – MG – Brasil

{lizardo, mirella, clodoveu}@dcc.ufmg.br

Resumo. *Hoje, dados geoespaciais são criados, armazenados e utilizados em uma quantidade nunca vista antes. No entanto, o grande volume de dados obtidos com sensores geográficos, satélites, redes sociais e demais serviços de localização tornou-se um desafio para os tradicionais bancos de dados relacionais e objeto-relacionais. Nesse cenário, gerenciadores de bancos de dados não relacionais, conhecidos como NoSQL, podem apresentar soluções mais eficientes para manipular grandes volumes de dados. Este trabalho propõe a construção de um protótipo NoSQL espacial, chamado GeoNoSQL, utilizando o Apache Cassandra, um gerenciador não relacional de alta escalabilidade e desempenho. Os recursos de indexação espacial são construídos utilizando a biblioteca de recuperação de informação Apache Lucene. Uma análise experimental de desempenho mostra que essa solução é superior ao PostGIS, a extensão espacial do PostgreSQL.*

Abstract. *Geospatial data are created, stored and used today in an unprecedented rate. However, the large volume of data collected from geographic sensors, satellites, social networks and other location services has become a challenge for relational database management systems. In this scenario, non-relational database management systems, known as NoSQL, can provide more efficient solutions to handle big volumes of data. This paper proposes the construction of a spatial NoSQL prototype, called GeoNoSQL, using Apache Cassandra, a non-relational database management system with high scalability and performance. The spatial indexes are built using the library for information retrieval Apache Lucene. Our experiments show that GeoNoSQL performs better than PostGIS, the spatial extension for PostgreSQL.*

1. Introdução

Os tradicionais Sistemas de Gerenciamento de Banco de Dados Relacionais (SGBD-R) têm sido, por anos, a principal solução de armazenamento de diferentes tipos de dados. No entanto, grandes volumes de dados geoespaciais são criados, armazenados e utilizados atualmente como nunca foram antes. Nesse cenário, gerenciadores de bancos de dados não relacionais (NoSQL) podem ser, por suas características, soluções mais eficientes para as aplicações que utilizam dados geoespaciais de grande volume.

A principal característica dos bancos de dados não relacionais é a sua capacidade de manipular grandes quantidades de dados sem a necessidade de definição de um esquema e ainda com alto desempenho. Por isso nesse trabalho propomos a construção de um banco de dados espacial utilizando o Apache Cassandra[1]

[1]Apache Cassandra: http://cassandra.apache.org/

[Lakshman and Malik 2010]. Esse banco de dados não relacional possui como características principais a escalabilidade e a alta disponibilidade sem ponto de falha e sem perda de performance.

O Apache Cassandra não possui mecanismos ou extensões para trabalhar com dados geoespaciais e por isso não é capaz de indexar dados multidimensionais. Porém, sabemos que o índice geoespacial pode aumentar significativamente o desempenho das consultas sobre dados geoespaciais. Face a esse impasse, o GeoNoSQL utiliza a biblioteca de recuperação de informação Apache Lucene[2] [Hatcher and Gospodnetic 2004] com a extensão geoespacial Lucene 4 Spatial[3] para construir o índice espacial. O desempenho do GeoNoSQL é avaliado em dois problemas típicos de sistemas de informação geográficos (SIG). O primeiro problema é a procura por polígonos que contêm um determinado ponto, e o outro é o problema dos vizinhos mais próximos. Em ambos, o GeoNoSQL é comparado ao PostGIS[4] [Obe and Hsu 2011], uma extensão espacial do SGBD relacional PostgreSQL.

O restante do artigo está organizado da seguinte forma. A Seção 2 apresenta conceitos e descreve trabalhos relacionados. A Seção 3 descreve a solução proposta, avaliada experimentalmente na Seção 4. A Seção 5 traz conclusões e lista trabalhos futuros.

2. Trabalhos Relacionados

Nessa seção são descritas as diferenças entre um SGBD-R e um NoSQL, e são listadas as características principais dos componentes envolvidos na solução proposta.

O termo NoSQL foi usado pela primeira vez por [Strozzi 1998] ao referir-se a um SGBD relacional que não oferecia suporte à linguagem de consulta SQL. Porém esse termo é hoje utilizado para designar os SGBDs não relacionais. Esses novos sistemas de armazenamento surgiram como uma alternativa aos sistemas tradicionais. Das suas características podem ser citadas o alto desempenho, a escalabilidade, a replicação e o suporte a dados estruturados como as mais interessantes [Agrawal et al. 2008].

Nos bancos NoSQL, não há relacionamentos entre as tabelas como nos sistemas relacionais, e os dados não precisam ter um esquema fixo. Existem diferentes SGBDs não relacionais e eles são classificados de acordo com a forma em que trabalham com os dados, a saber: *Column Store/Column Families*, *Document Store*, *Key Value/Tuple Store*, *Eventually Consistent Key Value Store*, *Graph Databases*, *Object Databases*, *Grid Database Solutions* e *XML Databases* [Tudorica and Bucur 2011].

Dentre as várias possibilidades de SGBDs NoSQL, o Cassandra é utilizado nessa solução. Ele foi desenvolvido inicialmente pelo Facebook, que abriu seu código em 2008 [Lakshman and Malik 2010]. Seu projeto e implementação compartilham estratégias presentes nos SGBDs relacionais, mas não oferecem suporte a um modelo de dados totalmente relacional e, sim, a um modelo simples com controle dinâmico sobre o layout e formato dos dados.

O modelo de dados do Cassandra consiste de uma tabela hash multidimensional distribuída em nós e estruturada pelas seguintes dimensões:

[2]Apache Lucene: http://lucene.apache.org/

[3]Lucene 4 Spatial: https://builds.apache.org/job/Lucene-Artifacts-4.x/javadoc/spatial/

[4]PostGIS: http://postgis.net/

- **Columns (colunas):** São constituídas pela tripla: nome, valor e *timestamp*. O Cassandra também oferece suporte às chamadas Supercolumns que possuem nome e outras colunas associadas;
- **Rows (linhas):** São indexadas por uma chave e as operações sobre elas são atômicas por réplica. Cada linha é constituída por diferentes colunas que são ordenadas conforme a especificação do cliente, seja pelo nome ou pelo *timestamp*;
- **Column Families (famílias de colunas):** São coleções de linhas rotuladas por um nome. Assemelham-se às tabelas dos bancos de dados relacionais, porém flexíveis, já que permitem um número arbitrário de linhas e colunas;
- **Keyspaces:** Agrupamento lógico de familias de colunas que fornecem um escopo para nomes.

A arquitetura do Cassandra particiona e distribui os dados entre os vários nós utilizando uma função hash consistente. Esse particionamento garante o balanceamento dos dados entre os nós, mas também pode ser especificado para garantir a ordem das linhas pelas chaves de forma a permitir consultas sobre intervalos de dados. A replicação dos dados entre os nós pode ser especificada pelo cliente conforme a definição do fator de replicação. Esses fatores permitem ao Cassandra trabalhar com grandes volumes de dados com eficiência. Esses foram os principais motivos da escolha do Cassandra para a utilização na solução proposta.

Uma vez definido o SGBD, é necessário escolher um índice que calcule as operações espaciais. Entre as possibilidades existentes, o Apache Lucene apresenta-se como uma biblioteca *open source* avançada para recuperação de informação. O foco principal do Lucene é a indexação e busca de textos para diversas aplicações como clientes de e-mail, procura de documentos, busca na Web, entre outros. Sites como Wikipédia, TheServerSide, jGuru e LinkedIn foram desenvolvidos com essa biblioteca. Disponível para várias linguagens de programação, dentre elas Java, Perl, Python, C++ e .Net, o Lucene é altamente escalável e possui vários recursos, como:

- Algoritmos de busca precisos e eficientes;
- Cálculo de pontuação para ranqueamento de relevância dos documentos buscados;
- Possibilita diferentes tipos de consultas, como consulta frasal, consulta por caractere curinga, busca com imprecisão, procura por valores numéricos e outros;
- Suporte a análise, classificação e filtragem de expressões de consultas;
- Implementa controle de concorrência no índice e permite a procura e a indexação de forma simultânea.

O Lucene possui uma extensão geoespacial chamada de Lucene 4 Spatial que adiciona suporte à indexação e pesquisa de objetos geoespaciais. Com essa extensão é possível indexar pontos, retângulos e círculos e ainda utilizar algumas funções espaciais como interseção espacial de dois objetos ou cálculo de distância.

Finalmente, [Malone, M. 2010] utilizou o Apache Cassandra para construir um banco de dados geoespacial [Finley, K. 2011], porém tal solução utiliza o próprio Cassandra como índice, enquanto o GeoNoSQL utiliza o Lucene. Conforme será mostrado na avaliação experimental, ter um índice eficiente é essencial para o desempenho de consultas geoespaciais.

3. Persistência e Consulta em GeoNoSQL

Nesta seção estão descritos os detalhes da solução implementada e estão explicados como são realizados a persistência e a consulta dos dados no banco de dados geoespacial proposto. Esse sistema associa as qualidades notáveis do Cassandra, como escalabilidade e desempenho de leitura e escrita, com a eficiência do índice Lucene. Para que um objeto possa ser armazenado nele é necessário que ele tenha uma *chave* que seja única, uma *geometria* que pode ser do tipo ponto, linha, polígono ou círculo e outros dados que o caracterize.

No GeoNoSQL, todos os objetos são armazenados no Cassandra, que permite recuperá-los com ótimo desempenho por meio de consultas diretas as suas chaves. Os retângulos envolventes dos objetos e suas coordenadas são indexadas no Lucene, que, com o módulo Lucene 4 Spatial, apresenta toda a estrutura necessária para indexar e realizar operações espaciais sobre essas entidades. Dessa forma, o papel do Lucene nessa solução é, durante uma consulta, calcular objetos candidatos e retornar, com eficiência, suas chaves para que estas possam ser utilizadas na recuperação dos dados no Cassandra. Posteriormente, esses objetos devem validados em memória para evitar falsos positivos, conforme será melhor explicado na Subseção 3.2.

3.1. Persistência

A persistência de novos dados no sistema é realizada em duas fases. Na primeira, o objeto tem sua geometria e demais dados indexados no índice do Lucene. Se a geometria do objeto for um ponto ou um círculo, ela própria é adicionada ao índice. No caso de ser uma linha ou um polígono, seu retângulo envolvente mínimo é indexado. Na segunda fase, os dados são armazenados no Cassandra em uma nova linha (cuja chave é a mesma do objeto) que é adicionada à família de colunas do banco de dados. As demais informações do objeto, incluindo geometria, são armazenadas como colunas dessa linha.

3.2. Consulta

A consulta de dados é realizada em três fases: *busca no índice Lucene*, para recuperar as chaves dos possíveis objetos que atendem à solução do problema; *requisição dos dados ao Cassandra*, para carregar para a memória os objetos cujas chaves foram apontadas pelo Lucene; e *validação*, para excluir possíveis objetos falsos positivos.

Na primeira fase, é enviada ao Lucene uma requisição com as características dos objetos que devem ser pesquisados no índice. Essa requisição permite filtrar os objetos por alguma característica específica e também por sua geometria. No sistema implementado, a requisição pode ser um dos seguintes casos:

1. **busca pelo vizinho mais próximo** – recupera objetos que estão a uma certa distância de um ponto dado. Os objetos são retornados em ordem crescente da distância geográfica ao ponto;
2. **busca por polígonos ou círculos envolventes** – recupera objetos do tipo polígono ou círculo que contêm um ponto dado;
3. **busca por pontos contidos** – recupera objetos do tipo ponto contidos em um polígono ou círculo dado.

Como resultado da primeira fase, obtém-se uma lista das chaves dos objetos encontrados pelo Lucene. Na segunda fase, essa lista é enviada ao Cassandra a fim de

recuperar os dados relacionados com as respectivas chaves. Nos casos 2 e 3, é necessário validar os objetos recuperados, pois a recuperação no índice é feita utilizando o retângulo envolvente mínimo dos objetos, e portanto pode retornar falsos positivos (ou seja, objetos dentro do retângulo envolvente, mas fora do polígono). Nesse caso, a validação (terceira fase do processo de consulta) é feita aplicando um algoritmo geométrico que verifica se um ponto está dentro de um polígono. Ele é aplicado para cada polígono ou círculo retornado no caso 2, e para cada ponto recuperado no caso 3.

4. Avaliação Experimental

O desempenho de consulta do GeoNoSQL foi avaliado em dois problemas. O primeiro é a procura por polígonos que contêm um determinado ponto, e o segundo é o problema de vizinhos mais próximos, conforme detalhados nas subseções 4.1 e 4.2. Para servir de comparação, foi utilizado o PostGIS, uma extensão geográfica de código aberto do PostgreSQL. Em ambos os problemas, os dados foram indexados no PostGIS utilizando índices espaciais genéricos para aumentar a perfomance das consultas.

Todos os testes foram executados em uma máquina rodando o Ubuntu 12.10 com processador Intel Core I7-3630QM, 2.40GHz x 8, 8GB de memória principal. Cada teste foi executado cinco vezes, descartando a primeira execução e calculando a média dos resultados das outras quatro execuções.

4.1. Problema 1: Procura por polígonos

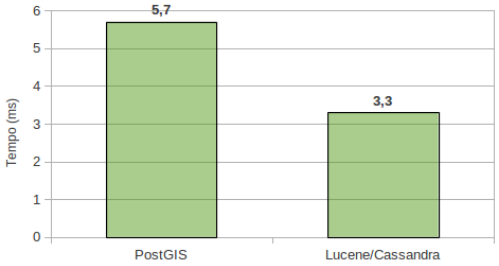

Figura 1. Desempenho das soluções para o timezone look-up.

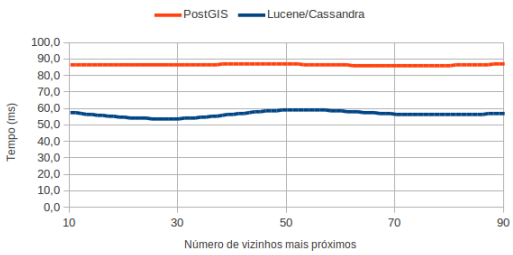

Figura 2. Desempenho das soluções para o problema dos vizinhos mais próximos.

Nesse experimento, foi utilizada uma base de dados contendo polígonos e multi-polígonos que representam 408 *timezones* do mundo[5]. O objetivo desse problema é, dado uma coordenada geográfica qualquer do planeta, identificar a qual *timezone* ela pertence. Na prática o objetivo é identificar qual polígono ou multi-polígono geográfico contém a coordenada pesquisada.

Os resultados dessa avaliação são apresentados na Figura 1 e representam os tempos médios da busca de 228 pontos aleatórios sobre a superfície do planeta incluindo oceanos.

Para comparação, no PostGIS, foi realizada, para cada um dos 228 pontos, a consulta seguinte:

[5]Fonte: Um arquivo *shape* contendo as timezones: http://efele.net/maps/tz/world/

```
SELECT timezone_name
FROM tz_world
WHERE ST_Contains(the_geom, ST_Point( Lon, Lat ));
```

Conforme os resultados apresentados, a solução proposta Lucene/Cassandra apresentou um tempo médio de busca inferior ao do PostGIS, sendo aproximadamente $2,4ms$ mais eficiente. Na Tabela 1, estão disponíveis os tempos médios gastos em cada uma das fases da consulta. A busca no índice do Lucene é a fase mais lenta desse processo.

4.2. Problema 2: Vizinhos mais próximos

Nesse problema, o desempenho do GeoNoSQL é testado para o problema dos vizinhos mais próximos. Considerando uma base de dados com objetos geográficos, o objetivo desse problema é identificar os objetos que estão próximos a uma coordenada de interesse em ordem crescente de distância. Para isso, foi utilizado um banco de dados contendo 201 mil estações de metrôs, trens e ônibus da Alemanha.

Os resultados obtidos nessa avaliação são apresentados na Figura 2 e representam os tempos médios da busca de 60 coordenadas geográficas de cidades alemãs. O parâmetro que determina o número de vizinhos mais próximos a serem retornados também foi analisado para verificar se há perda de desempenho com seu aumento, já que mais pontos deverão ser calculados e ordenados pelo índice.

A consulta equivalente no PostGIS é a seguinte:

```
SELECT id, place_name
FROM tt_germany
ORDER BY the_geom <-> ST_Point( Lon , Lat )
LIMIT x;
```

Os resultados apresentados mostram que a solução proposta foi novamente superior ao PostGIS, com um tempo médio $30ms$ mais rápido. Ao contrário do que era esperado, não houve perda de performance com o aumento do número de vizinhos a serem retornados. Nesse tipo de problema, não há a necessidade de fazer uma verificação ao final, pois não há a possibilidade de existirem falsos positivos. Conforme listado na Tabela 1, a parte lenta do processamento foi a busca no índice, representando quase 95% do tempo.

Tabela 1. Tempo médio gasto em cada fase da consulta.

		Problema 1	Problema 2
Fase 1	Busca no índice (Lucene)	79,92%	94,66%
Fase 2	Recuperação no banco de dados (Cassandra)	19,52%	5,33%
Fase 3	Algoritmo de ponto em polígono	0,56%	-

5. Conclusão

Este trabalho propôs a construção de um banco de dados espacial utilizando o banco de dados NoSQL Apache Cassandra. O objetivo do trabalho foi utilizar um sistema cujas características principais são alta escalabilidade e alto desempenho, para armazenar e manipular dados espaciais que exigem sistemas mais robustos pelo fato dos dados serem de

grande volume. A extensão espacial *Lucene 4 Spatial* da biblioteca de recuperação de informação Apache Lucene foi utilizada para construir um índice espacial, já que o Cassandra não possui suporte a dados desse tipo. Na avaliação experimental, o GeoNoSQL apresentou resultados satisfatórios com desempenho superior ao PostGIS, uma extensão espacial para o banco de dados relacional PostgreSQL. No futuro, pretende-se testar essa solução em um ambiente de teste distribuído, no qual o Cassandra terá mais nós, que podem aumentar ainda mais o desempenho.

Agradecimentos

Os autores registram e agradecem ao CNPq e à FAPEMIG pelo apoio a suas atividades de pesquisa.

Referências

Agrawal, R., Ailamaki, A., Bernstein, P. A., Brewer, E. A., Carey, M. J., Chaudhuri, S., Doan, A., Florescu, D., Franklin, M. J., Garcia-Molina, H., et al. (2008). The claremont report on database research. *ACM SIGMOD Record*, 37(3):9–19.

Finley, K. (2011). Video: How simplegeo built a scalable geospatial database with apache cassandra. Available at: http://www.readwriteweb.com/cloud/2011/02/video-simplegeo-cassandra.php (09.07.2013).

Hatcher, E. and Gospodnetic, O. (2004). *Lucene in Action (In Action series)*. Manning Publications Co., Greenwich, CT, USA.

Lakshman, A. and Malik, P. (2010). Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44(2):35–40.

Malone, M. (2010). Presentation: Scaling gis data in non-relational data stores. Available at: http://strangeloop2010.com/talks/14495 (09.07.2013).

Obe, R. and Hsu, L. (2011). *PostGIS in action*. Manning Publications Co.

Strozzi, C. (1998). NoSQL: a non-sql rdbms. *Online verfügbar unter: http://www. strozzi. it/cgi-bin/CSA/tw7/I/en US/nosql/Home% 20Page, abgerufen am*, 25:2012.

Tudorica, B. G. and Bucur, C. (2011). A comparison between several NoSQL databases with comments and notes. In *Roedunet International Conference (RoEduNet), 2011 10th*, pages 1–5. IEEE.