NoSQL (Cassandra)

Conceitos e características

Erick Haendel, Luís Carlos, Rodrigo Félix

Departamento de Estatística e Informática - Universidade Federal Rural de Pernambuco (UFRPE)

Rua Dom Manoel de Medeiros, s/n, Dois Irmãos - CEP: 52171-900 - Recife/PE

{erickhaendel, luis.c.ferraz, rodrigofelixrodrigo}@gmail.com

Resumo. *Este artigo trás uma abordagem das principais características do NoSQL com foco no tipo Família de colunas, aqui descrito pelo Apache Cassandra. O NoSQL tem na escalabilidade seu ponto forte, o que lhe atribui uma grande velocidade e capacidade de armazenar dados. Esta tecnologia vem sendo bastante utilizada pelas principais empresas da internet, como o Facebook, Google e Twitter.*

1. Introdução

Desde a década de 80, o modelo relacional impera praticamente absoluto entre os sistemas de gerenciamento de banco de dados. Se tornou uma ferramenta tão poderosa e confiável que a preocupação dos desenvolvedores se restringia a modelagem do esquema de dados. Sua força e consolidação no mercado se deve, também, a linguagem SQL, por sua simplicidade e seu alto poder de expressão nos bancos relacionais.

Mesmo resistindo a evolução da tecnologia da informação, a crescente das aplicações web 2.0, aumento de usuários online, computação em nuvem, as bases de dados cresceram a níveis exponenciais, e o modelo relacional já não pode oferecer suporte com a mesma eficiência de outrora.

Para suprir essa necessidade, empresas como a Google e a Amazon desenvolveram novas soluções, baseadas em modelos de dados não relacionais, para que pudessem oferecer maior escalabilidade e maior disponibilidade independendo do volume de dados.

Esses novos sistemas foram batizados de NoSQL.

2. O NoSQL

Not Only SQL (não apenas SQL) surgiu com a importância em se obter altos graus de paralelismo no processamento dos grandes volumes de dados e a possibilidade de distribuição destes dados em escala global. Sistemas relacionais apresentam complexidade superior e altos custos para obedecer tais requisitos de operação. Já os sistemas não relacionais possuem estes requisitos como alguns de seus atributos principais, que são:

- **Escalabilidade horizontal:** Possibilidade de adição de nós ao sistema (novos servidores).
- **Replicação:** Criação de cópias em outros ambientes para aumentar a disponibilidade. Possui dois tipos de arquitetura para realizar as operações:

  MASTER X SLAVE: Cada escrita em banco resulta em X escritas, onde X é o número de slaves. Sendo assim, temos um banco "Master" que propaga cada escrita para os bancos "slaves". Possibilita aumento na velocidade de leitura, mas não melhora a capacidade de escrita. Caso o "Master" falhe, a capacidade de escrita fica comprometida. Possui sincronia eventual entre os "slaves", ou seja, não utilizam alto tráfego de rede.

  MASTER X MASTER: Aumentamos o número de "Masters" em nosso sistema e assim aumentamos nossa capacidade de escrita. Apresenta alto tráfego na rede por sincronização constante entre os bancos "Master".
- **Schema-Free:** Um dos fatores que possibilitam a capacidade de escalabilidade é a ausência de esquema.
- **Clusterização:** Consiste num banco de dados armazenado e gerenciado por mais de um servidor de alto desempenho e alta disponibilidade.
- **Mapreduce:** Algoritmo patenteado do Google, que divide os dados no nó inicial para serem processados por outros nós. Estes dados processados retornam ao nó principal que combina as respostas gerando o resultado final.
- **Sharding:** Consiste em dividir os dados horizontalmente, ou seja, quebrar as tabelas, diminuindo seu número de linhas e separando-as em ambientes diferentes.

Para atingir esse potencial de escalabilidade, os sistemas NoSQL não obedecem as regras ACID (consistência forte), sendo inclinados aos conceitos BASE (consistência fraca), ou seja, as garantias do modelo relacional foram reduzidas para se obter maior potencial de escalabilidade. O teorema de CAP, de Eric Brewer, formalizou essas observações da seguinte forma: "sistemas escaláveis, tolerância a partição, disponibilidade e consistência estão em relação de compensação, sendo impossível desenvolver sistemas contendo todas estas propriedades."

3. Os principais tipos de bancos de dados NoSQL

- **Chave-valor**

Baseados no Dynamo da Amazon, esse tipo de banco de dados tem foco na escalabilidade para grandes volumes de dados. Consiste em pares "chave (única) – valor" e armazenam objetos indexados por chaves, possibilitando buscas por esses objeto a partir de suas chaves.

Exemplos: RIAK, Redis e MemcacheDB.

- **Orientado a documentos**

Armazenam e organizam os dados como coleções de documentos, ou seja, coleções de atributos e valores. Em geral, os bancos de dados orientados a documento não possuem esquema, ou seja, os documentos armazenados não precisam possuir estrutura em comum.

Exemplos: MongoDB e CouchDB.

- **Família de colunas**

Esse tipo de banco de dados tornou-se popular com o BigTable do Google. Possui organização por colunas (unidade básica) e utiliza chave única. Apresenta flexibilidade em comparação ao modelo relacional, sem poluir as linhas com colunas nulas.

A coluna é uma tupla no padrão (nome, valor, timestamp).

Exemplos: Cassandra, HBase e Hypertable.

- **Orientado por grafos**

Apresenta foco na modelagem de estrutura dos dados. Representa os dados como grafos dirigidos ou como estruturas que generalizem a noção de grafos. Operações sobre os dados são transformações sobre o grafo, e fazem uso de conceitos de grafos, como caminhos, vizinhos e sub-grafos.

Inspirado no teorema matemático dos grafos (G=(E,V)).

Exemplos: Neo4j e InfoGrid.

4.Cassandra

O Cassandra foi desenvolvido pelo Facebook, para ajudar no funcionamento de sua caixa de busca e armazenar/vasculhar todas as mensagens das caixas de entrada dos usuários. Em julho de 2008, ele foi liberado sob licença Open Source. Foi colocado como um projeto de incubação da Fundação Apache em janeiro de 2009. Em março de 2010 a primeira versão (0.3) é lançada, estando atualmente na versão 1.2.3. Implementado em Java, ele foi fortemente influenciado pelo Dynamo, da Amazon, pioneiro na criação de um banco de dados do tipo chave/valor. Cassandra implementa um modelo de replicação de dados parecido com o Dynamo – sem nenhum ponto único de falha. Porém a forma de armazenar os dados é mais semelhante ao Google Bigtable (orientado a colunas).

O Cassandra tem excelentes características técnicas como ser facilmente escalável, eventualmente consistente, durável e tolerante a falhas. Seu poder de armazenamento chega a centenas de terabytes de dados, tendo suporte para replicação em vários datacenters.

4.1. O modelo de dados

O modelo de dados do Cassandra foi projetado para dados distribuídos em uma escala muito grande, na qual milhares de dados são distribuídos e replicados ao longo de várias máquinas, nas quais operam em conjunto de forma que aparecem como um única instância para o utilizador final. Desta forma, o Cassandra não é a melhor escolha caso se deseje executar um único nó.

Para desenvolvedores e administradores que vêm do mundo relacional, o modelo de dados do Cassandra pode ser inicialmente muito difícil de compreender. Palavras como "keySpace", são novas, e algumas, como "coluna", existem em ambos os modelos de dados (BD Relacional, BD NoSQL), mas com significados diferentes.

Em uma base de dados relacional, o banco de dados contém tabelas. Tabelas tem nome e contém uma ou mais colunas, que também tem nomes. Quando adicionamos dados a uma tabela, e especificamos os valores para cada coluna definida, se temos um valor vazio para uma coluna usamos nulo. Esta nova entrada adiciona uma linha à tabela, que posteriormente será lida. O Cassandra age de forma diferente, adiciona-se coluna para cada dado inserido, então, as linhas terão tamanho diferentes, constituídas somente de dados enviados.

Vamos começar a falar sobre a estrutura do modelo de dados do menor nível ao maior, após cada ponto abordado se tem como base para exemplificar visualmente a figura 1.

Coluna (Column) :

Esta é a unidade mais básica do modelo de dados do Cassandra, contém um nome, um valor e um registro de data e hora. No modelo relacional, se define a estrutura das tabelas ao criar o esquema do banco, já atribuindo todas as colunas e deixando para depois apenas a inserção dos dados, ou seja, fornecendo valores para uma estrutura pré-definida. No Cassandra, você não define primeiramente as colunas e sim a família de colunas que deseja na keyspace (1.4), sendo assim, podemos criar as colunas que desejamos a medida que vamos inserindo os dados, otimizando o espaço utilizado.

Super Coluna (Super Columns):

É uma coluna que armazena um conjunto de colunas. Este método de 'Colunas' dentro de outra coluna não pode passar de um nível.

Os desenvolvedores recomendam não usar supercolunas.

Linha (Row):

É uma coleção de colunas identificadas com um nome, o tamanho da linha é variável de acordo com os dados de entrada. Em uma 'família de colunas' pode haver várias linhas com tamanhos diferentes, isso é uma abordagem totalmente diferente do ER onde todas as linhas tem o mesmo tamanho, sendo que com valores nulos ou não.

Exemplo de uma linha:

```
"Second Foundation"-> { author="Asimov",  publishedDate="..", tag1="sci-fi", tag2="Asimov"    }
```

Família de Colunas (Column Families):

É uma coleção de linhas identificadas por um nome. A família de colunas tem a estrutura parecida com a de uma tabela no ER.

Exemplo de uma família de colunas:

```
Books->{
"Foundation"->{author="Asimov", publishedDate=".."},
"Second Foundation"->{author="Asimov", publishedDate=".."},
…
}
```

Keyspace :

É um grupo de várias famílias de colunas juntas. É apenas um agrupamento lógico de famílias de colunas e fornece um escopo isolado para nomes.

Na figura 1 é ilustrado um keyspace, ele pode ser entendido como o nome do banco de dados em um modelo relacional. No Cassandra o keyspace é o recipiente mais externo de dados.

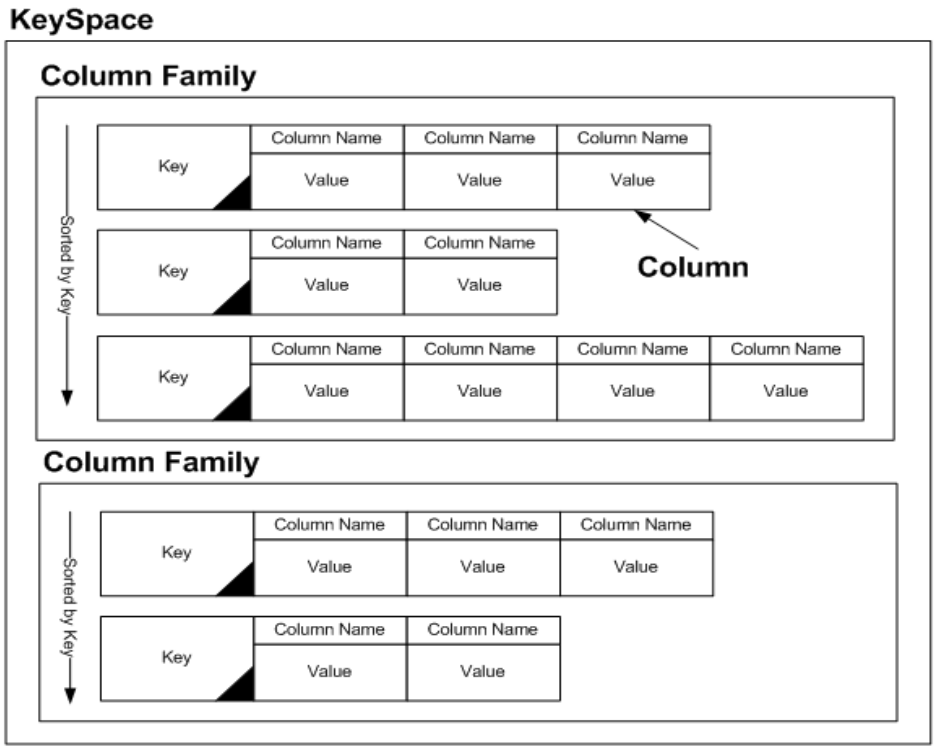

Figura 1. Modelo de dados

Cluster :

Um grupo de nós onde se armazenar os dados. Pode-se criar um cluster de nó único. Um nó é entendido como um novo servidor adicionado à rede.

Replicação:

Ao criar as keyspace pode-se declarar o fator de replicação, que determina quantas cópias dos dados poderão existir em outros nós da rede.

As vantagens da replicação são:

- **Maior disponibilidade:** os dados permanecem acessíveis diante de falhas;
- **Maior segurança:** dados redundantes.

E a principal desvantagem:

- **Controle de consistência**: é preciso manter as réplicas atualizadas.

Ao implementar uma replicação, deve-se levar em conta o que vale mais a pena para o projeto em questão, o desempenho ou a consistência, visto que uma grande replicação pode causar certa lentidão no tráfego dos dados, mas também aumenta a

segurança por ter o mesmo dado em outros nós, não deixando o sistema cair durante uma falha em um nó.

Particionador:

Uma estrutura interna do Cassandra que determina qual dos nós irá armazenar os dados e isso acontece de modo que essa distribuição ocorra uniformemente entre os nós do cluster para balanceamento de carga.

O Cassandra faz um particionamento dos dados de forma transparente nos nós que participam da database no cluster, assim, cada nó fica responsável por uma parte da database global.

A escolha de onde serão colocados os dados no nodo é com base na row key da column family.

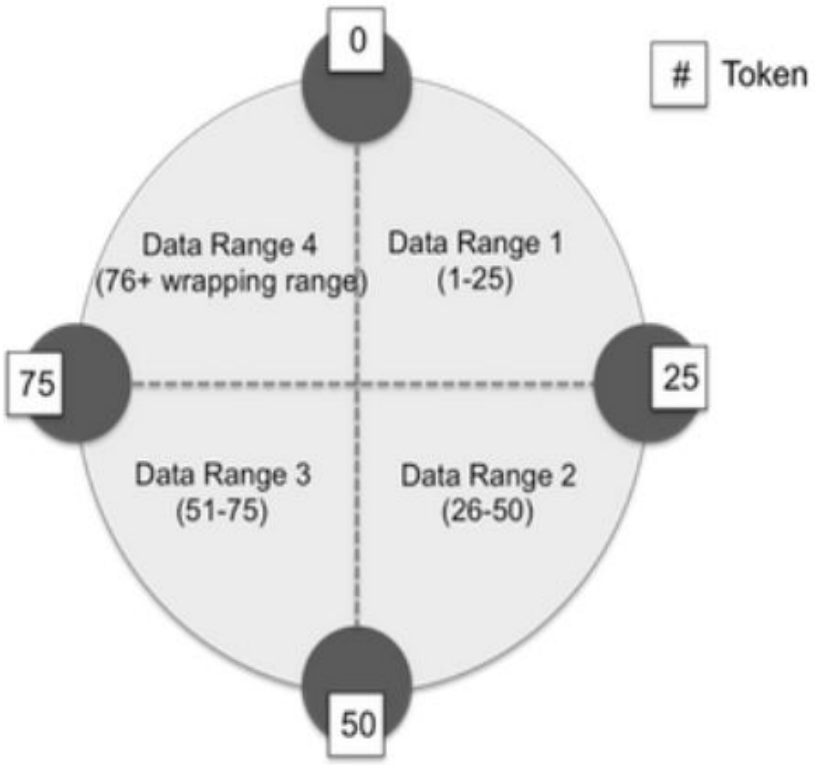

Figura 2. Anel

O número total de dados armazenados pelo cluster é representado pelo anel (figura 2):

- Dividido em intervalos iguais ao número de nós;
- Cada nó pode ser responsável por um ou mais intervalos de dados.

Um nodo pode-se juntar a um anel, onde lhe é atribuído um token, que é criado por uma ferramenta interna de geração de token, sendo este token indispensável antes do nós começarem a trabalhar pela primeira vez.

Este define:

- Posição do nodo no anel;
- Faixa de dados.

Para determinar onde o nodo irá ficar no anel:

● Em sentido horário se localiza o nodo com valor de chave maior do que a chave do nodo a entrar (ou da "row key").

● Cada nó é responsável pela região entre si e de seu antecessor.

● No exemplo (figura 2), nodo ZERO de 75 a 0.

4.2.Possíveis surpresas com o Cassandra

Como nada é perfeito, existem algumas desvantagens no uso do Cassandra ou na transação de um ER para ele:

- Sem transações, sem JOINs
- Sem chaves estrangeiras. As chaves são imutáveis:
- As chaves devem ser exclusivas:
- A procura é complicada
- Recomenda-se não usar supercolunas e particionadores de preservação
- Documentação escassa;
- Pode ocorrer um desconforto para o programador se adaptar ao modelo;
- Incompatibilidade com modelo relacional.

5. Referências

Cadmin Tool

(http://cadmintool.blogspot.com.br/2012/09/o-que-e-nosql.html)

IBM developerworks - Srinath Perera

(http://www.ibm.com/developerworks/br/library/os-apache-cassandra/)

iMasters - Jean Nascimento (http://imasters.com.br/artigo/17043/banco-de-dados/nosql-voce-realmente-sabe-do-que-estamos-falando/)

Cassandra – Explorando o Cassandra

(http://cassandraufg.wordpress.com)

NoSql - Arthur Azevedo - Rafael Benedito
(http://pt.scribd.com/doc/73007407/6/Principais-Caracteristicas-NoSQL)