Cassandra – Uma sistema de armazenamento NoSQL altamente escalável

Tiago Pasqualini da Silva
Universidade Federal de São Carlos – *Campus* Sorocaba
Sorocaba, São Paulo
Email: tiago.pasqualini@gmail.com

***Resumo*—O grande crescimento no número de acessos em diversos serviços oferecidos na Internet requer que os dados sejam armazenados de forma eficiente e escalável. O Cassandra, um sistema de banco de dados NoSQL baseado em chave/valor afirma garantir um armazenamento de dados distribuído, altamente escalável e eventualmente consistente. Com o seu uso, várias empresas otimizaram seus serviços, pois o Cassandra se mostra bem mais rápido que as outras soluções de banco de dados distribuídos baseados em NoSQL.**

***Palavras-chave* —** cloud computing, banco de dados, sistemas distribuídos, NoSQL, escalabilidade, disponibilidade.

I. Introdução

Hoje em dia, grandes serviços vêm sendo oferecidos através da Internet. Esses grandes serviços possuem um número absurdo de acessos simultâneos e devem manter-se disponíveis o tempo todo. Para garantir essa disponibilidade, grandes empresas como Google e Facebook apostam no novo modelo de computação chamado de Cloud Computing. Um dos fatores chave nesse novo modelo de computação é o armazenamento dos dados.

O armazenamento de dados, que antes era feito em bancos de dados relacionais comuns, hoje em dia se tornou obsoleto para alguns tipos de serviços que precisam guardar volumes enormes de dados com rapidez e eficiência. Como solução para esses serviços surgiram os bancos de dados NoSQL [2]. Nessa nova abordagem, os bancos de dados se tornaram mais simples, porém mais robustos. Além disso, esses bancos de dados NoSQL possuem, em sua grande maioria, tolerância ao particionamento de dados, o que ajuda muito na escalabilidade, pois os dados podem ser divididos entre diversas máquinas para dividir o processamento desses dados entre as várias máquinas.

Nessa linha, o Facebook tinha um problema com as suas buscas nas mensagens enviadas entre os usuários. O volume de dados que era enorme tinha a característica de crescer muito rapidamente e precisavam ser acessados com certa agilidade. Para solucionar esse problema, surgiu o Cassandra, que é um sistema de armazenamento distribuído e altamente escalável para dados estruturados.

Esse artigo tem como objetivo dar uma visão geral sobre o Cassandra, mostrar seu modelo de dados e explicar alguns detalhes de sua arquitetura. Na seção II é encontrada uma explicação das principais características do Cassandra, na seção III é explicado detalhes sobre o modelo de dados do Cassandra, ne seção IV é dada uma visão geral da arquitetura do sistema focando na partição e replicação dos dados, na seção V são apresentados três estudos de caso de empresas que adotaram o Cassandra em seus serviços e finalmente na seção VI temos a conclusão.

II. O Cassandra

O Cassandra é um sistema de banco de dados baseado na abordagem NoSQL. Existem alguns tipos diferentes de NoSQL, sendo que o Cassandra é baseado no tipo chave/valor. Nesse tipo de banco de dados, os dados são identificados através de uma chave.

A principal promessa do Cassandra é de prover um sistema de armazenamento distribuído, altamente escalável e eventualmente consistente. Para garantir essas promessas, foram unidas características de dois sistemas NoSQL, o BigTable [11] do Google e o Dynamo [12] da Amazon.

Esse sistema foi criado no Facebook em 2008, por Avinash Lakshman e Prashant Malik. Foi bastante usado no próprio Facebook para tornar a busca de mensagens mais robusta. No final de 2010 seu uso no Facebook foi descontinuado. Hoje em dia é utilizada uma outra solução NoSQL no lugar do Cassandra. Mais informações sobre o uso tanto no Facebook como em outros serviços pode ser encontrado na seção V.

III. Modelo de dados

No Cassandra, os dados são indexados por uma chave do tipo String. Essa chave referencia uma linha, onde os dados se encontram, sendo que em cada linha os dados são divididos em colunas e famílias de colunas. Essas divisões dos dados serão explicadas nessa seção.

A. Colunas

Cada coluna no Cassandra tem um nome, que a identifica, um valor e um timestamp, sendo que tanto o valor quanto o timestamp são fornecidos pela aplicação cliente quando um dado é inserido. Além disso, uma aplicação pode criar uma coluna em tempo de execução, sem precisar declarar ou alterar

nada, ou seja, basta informar um valor para essa nova coluna para uma dada chave.

Além das colunas normais, existem as super colunas (Super Columns), que são colunas que em vez de terem objetos como valores, possuem outras colunas. Por exemplo, é possível ter uma coluna chamada homeAddress que tem como valores as colunas street e city, sendo que tanto a coluna street como a coluna city possuem um valor e um timestamp. Abaixo temos na Figura 1 uma representação dessa tabela no formato JSON.

```
{   // this is a SuperColumn
    name: "homeAddress",
    // with an infinite list of Columns
    value: {
        // note the keys is the name of the Column
        street: {name: "street", value: "1234 x street", timestamp: 123456789},
        city: {name: "city", value: "san francisco", timestamp: 123456789},
        zip: {name: "zip", value: "94107", timestamp: 123456789},
    }
}
```

Figura 1: Representação dos dados de Super Colunas no formato JSON

B. Famílias de Colunas

Para agrupar as colunas, o Cassandra possui o conceito de Famílias de Colunas (Column Families), que é bem parecido com as tabelas de um banco de dados relacional. Diferentemente da colunas, as Column Families não são dinâmicas e precisam ser declaradas anteriormente em um arquivo de configuração.

Analogamente às colunas, também existem as Super Famílias de Colunas (Super Column Families), que são Famílias de Colunas que possui como colunas somente Super Colunas. Por exemplo, uma Super Família de Colunas chamada AddressBook, pode ter as colunas representando cada entrada na agenda de endereços. Dentro de cada coluna pode-se ter as colunas rua, zip e city, ou seja, para cada entrada na agenda, temos a rua do endereço, a cidade e o código postal. A representação desses dados no formato JSON pode ser encontrada na figura 2. Para melhor didática, os indicativos de nome, valor e timestamp das colunas foram retirados.

```
AddressBook = { // this is a ColumnFamily of type Super
    phatduckk: {   // this is the key to this row inside the Super CF
        // the key here is the name of the owner of the address book

        // now we have an infinite # of super columns in this row
        // the keys inside the row are the names for the SuperColumns
        // each of these SuperColumns is an address book entry
        friend1: {street: "8th street", zip: "90210", city: "Beverley Hills", state: "CA"},

        // this is the address book entry for John in phatduckk's address book
        John: {street: "Howard street", zip: "94404", city: "FC", state: "CA"},
        Kim: {street: "X street", zip: "87876", city: "Balls", state: "VA"},
        Tod: {street: "Jerry street", zip: "54556", city: "Cartoon", state: "CO"},
        Bob: {street: "Q Blvd", zip: "24252", city: "Nowhere", state: "MN"},
        ...
        // we can have an infinite # of ScuperColumns (aka address book entries)
    }, // end row
    ieure: {    // this is the key to another row in the Super CF
        // all the address book entries for ieure
        joey: {street: "A ave", zip: "55485", city: "Hell", state: "NV"},
        William: {street: "Armpit Dr", zip: "93301", city: "Bakersfield", state: "CA"},
    },
}
```

Figura 2: Representação dos dados de Super Famílias de Coluans no formato JSON

Finalmente, as Famílias de Colunas são agrupadas em Keyspaces. Esses Keyspaces podem ser comparados aos Schemas em um banco de dados relacional. A Figura 3 mostra um exemplo de banco de dados no Cassandra.

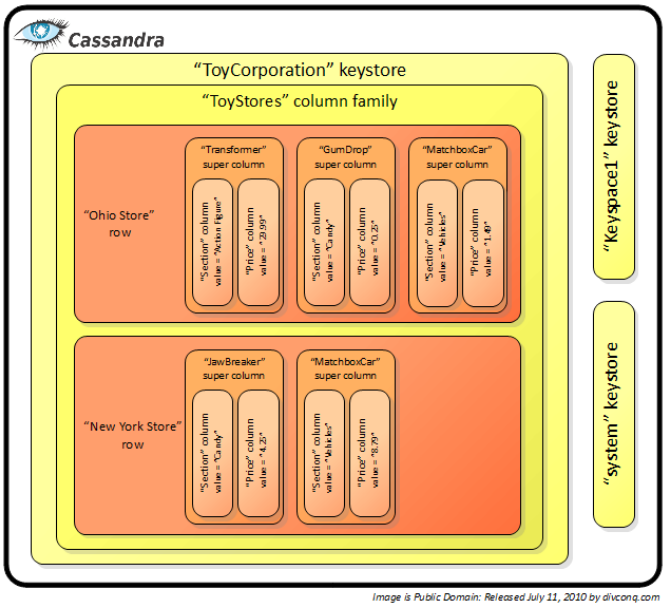

Figura 3: exemplo de banco de dados no Cassandra

IV. Arquitetura

Segundo o teorema de CAP[13], um sistema de banco de dados somente pode abranger dois itens entre consistência, disponibilidade e tolerância a partições. Entre essas três opções, o Cassandra garante a disponibilidade dos dados e a tolerância a partições.

Para garantir que um grande volume de dados seja tratado de forma rápida e eficiente, o Cassandra possui por trás uma arquitetura bastante robusta e complexa. Essa arquitetura é composta por diversos sistemas distribuídos que garantem que cada parte do sistema funcione corretamente.

Quando uma requisição de leitura ou escrita é feita, qualquer nó do cluster pode tratá-la. Através da chave, o nó que atendeu a requisição consegue saber quais nós possuem informações dos dados. O sistema então aguarda até que um número configurado de réplicas, chamado quorum, responder com o dado, no caso de leitura; ou responder com uma confirmação, no caso de uma escrita.

A. Particionamento dos dados

O cluster do Cassandra é organizado em um anel, sendo que cada nó do anel possui um intervalo de valores determinado. Os dados são divididos entre os diversos nós através de um hashing da chave identificadora desses dados. Esse hash gera um valor que então determina, através do intervalo de cada nó do anel, qual dessas máquinas será responsável por esse dado. Essas máquinas responsáveis pelos dados são chamadas de coordenadoras.

Essa divisão dos nós em anel facilita a adição e remoção de nós no cluster. Quando um nó é adicionado ou removido no cluster, somente os vizinhos desse nó são afetados. Isso garante que o sistema seja bastante escalável, já que a adição de um nó no cluster não afeta o funcionamento de todo o sistema.

B. Replicação dos dados

A replicação no Cassandra é utilizada para garantir a alta disponibilidade dos dados. Cada dado é encontrado em N nós do cluster, sendo que N é um fator de replicação configurável. O coordenador do nó (definido anteriormente) é o responsável por replicar os dados entre os N-1 nós. Essa réplica feita pelo coordenador pode ser feita de três formas diferentes: Rack Unaware, Rack Aware e Datacenter Aware. No modo Rack Unaware, o coordenador simplesmente seleciona os próximos N-1 nós do anel. Nos modos Rack Aware e Datacenter Aware, é utilizado um sistema externo que elege um líder, que tem como função avisar cada nó que se conecta ao cluster a faixa de valores do anel que ele será uma réplica. A figura 4 tem uma representação desse anel e mostra como os dados são gravados e replicados.

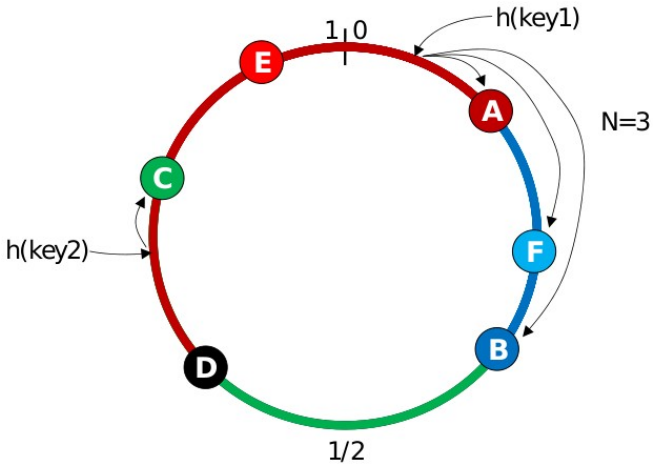

Figura 4: Exemplo de gravação de dados no Cassandra

V. Estudos de caso

Para demonstrar o uso do Cassandra em cenários reais, foram selecionados três estudos de caso de três grandes serviços oferecidos na Internet: o Facebook , o Twitter e o Reddit.

A. Facebook

O Facebook utilizou o Cassandra para agilizar seu novo sistema de busca de mensagens [6], lançado em 2008. O cluster utilizado tinha cerca de 600 núcleos, armazenando cerca de 150 terabytes de dados. Na época do uso, o sistema conseguia aguentar bem o uso da busca pelos 250 milhões de usuários da rede social.

Em novembro de 2010, uma nova versão do sistema de mensagens do Facebook foi lançado [7]. Durante o desenvolvimento dessa nova versão, foram testados três sistemas de armazenamento de dados diferentes: o já bastante conhecido e relacional MySQL, o Cassandra e o HBase, que também utiliza a abordagem NoSQL. Depois dos vários testes realizados, a equipe do Facebook optou por utilizar o HBase como seu novo sistema de banco de dados, por ser bastante escalável, ter um bom balanceamento de carga, ter boa performance, entre outros fatores.

B. Twitter

Além do Facebook, o Twitter [8] também passou a usar o Cassandra no lugar do MySQL. O uso, porém, não é no armazenamento dos tweets em si, mas sim em alguns serviços internos da rede social. Um dos exemplos de uso fornecidos pelo Twitter é em um sistema de análise de tweets por região para gerar os termos mais postados em uma certa região.

C. Reddit

O Reddit, um site de publicações sociais, recentemente teve um problema com seu sistema de cache [9]. Esse sistema utilizava o banco de dados memcachedb, que também é do tipo NoSQL. Porém, o banco de dados começou a apresentar problemas de escalabilidade e, para resolver esses problemas, a equipe do Reddit resolveu abandonar seu uso e migrar todo o sistema de cache para o Cassandra, por ser mais rápido e mais escalável que a solução anterior.

Depois de um tempo rodando essa nova solução utilizando o Cassandra, o Reddit começou a ter alguns problemas [10] com um sistema de cache anterior (que atende as requisições antes do Cassandra, ou seja, é um cache do Cassandra) o que levou a uma sobrecarga do Cassandra. A solução de adicionar mais nós para diminuir a carga não funcionaria devido a detalhes da inicialização do Cassandra, portanto foi necessário esperar a carga diminuir para que novas instâncias fossem iniciadas. Mesmo após esse problema, o Reddit continua a utilizar o Cassandra em seu sistema de cache.

VI. Conclusão

Com a leitura desse artigo e principalmente dos estudos de caso, pode-se perceber que o Cassandra é uma ótima solução para quem precisa de um sistema de armazenamento de dados altamente escalável. Suas características vão de encontro com as expectativas e necessidades de grandes serviços oferecidos na Internet, portanto, apesar de bastante nova, é uma solução bastante promissora.

Referências

[1] Avinash Lakashman, Prashant Malik, Cassandra – A Decentralized Structured Storage System, 2009.

[2] Renato Molina Toth, Abordagem NoSQL – Uma real alternativa, 2011

[3] Jonathan Lampe, Introduction to Cassandra Columns, Super Columns and Rows. Disponível em: http://www.divconq.com/2010/cassandra-columns-and-supercolumns-and-rows/

[4] Cassandra Wiki. Disponível em: http://wiki.apache.org/cassandra/

[5] Arin Sarkissian, WTF is a SuperColumn? An Intro to the Cassandra Data Model. Disponível em: http://arin.me/blog/wtf-is-a-supercolumn-cassandra-data-model

[6] Avinash Lakashman, Cassandra – A structured storage system on a P2P Network. Disponível em:

http://www.facebook.com/note.php?note_id=24413138919&id=9445547199&index=9

[7] Kannan Muthukkaruppan, The Underlying Technology of Messages. Disponível em: http://www.facebook.com/notes/facebook-engineering/the-underlying-technology-of-messages/454991608919

[8] Ryan King, Cassandra at Twitter Today. Disponível em: http://engineering.twitter.com/2010/07/cassandra-at-twitter-today.html

[9] David King, She who entangles men. Dispnível em: http://blog.reddit.com/2010/03/she-who-entangles-men.html

[10] Reddit, reddit's May 2010 "State of the Servers" report. Disponível em: http://blog.reddit.com/2010/05/reddits-may-2010-state-of-servers.html

[11] Fay Chang, Jeffrey Dean, Sanjay Ghemawat, Wilson C. Hsieh, Deborah A. Wallach, Mike Burrows, Tushar Chandra, Andrew Fikes, and Robert E. Gruber, Bigtable: A Distributed Storage System for Structured Data, 2006

[12] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall and Werner Vogels, Dynamo: Amazon's Highly Available Key-value Store, 2007

[13] Julian Browne, Brewer's CAP Theorem, 2009. Disponível em: http://www.julianbrowne.com/article/viewer/brewers-cap-theorem