Cassandra

Banco de Dados NoSQL

Geovani S. Amaral [1], Thales Maurício [1]

[1]Faculdade de Engenharia de computação – Universidade Federal do Pará (UFPA)

Rua Itaipu, 36 - Vila Permanente - 68464-000 - Tucuruí - PA - Brasil

{geovani.af4@gmail.com, thalesmauciq46@gmail.com}

Resumo: *Este trabalho tem como foco principal trazer uma abordagem no NoSQL e no banco de dados do Cassandra Apache. Com um grande aumento no número de acessos nos diversos serviços oferecidos na Internet, é necessário que os dados sejam armazenados de forma eficiente e escalável. Portanto, Cassandra, um sistema de banco de dados NoSQL baseado em chave/valor, afirma garantir um armazenamento de dados distribuído altamente escalável e, eventualmente consistente. O estudo mostra por que ele pode manipular grandes volumes de dados e apresenta algumas propriedades que o NoSQL usa para gerenciar informações.*

Abstract: *This work has as main focus to bring an approach in NoSQL and in the Cassandra Apache database. With a large increase in the number of accesses in various services offered on the Internet, it is necessary that the data be stored in an efficient and scalable. Therefore, Cassandra, a NoSQL database system based on key/value, claims to ensure a highly scalable and eventually consistent distributed data storage. The study shows why it can handle large volumes of data and presents some properties that NoSQL uses to manage information.*

1. Introdução

Nos dias atuais grandes serviços são oferecidos através da internet. Empresas conhecidas mundialmente, como Amazon, Facebook e Google, são responsáveis por ter que manipular grandes volume de dados e por operarem online constantemente, com isso eles desenvolveram suas próprias soluções de bancos de dados NoSQL [24]. O banco de dados Cassandra é uma banco pós-relacional, mas esse termo não é muito popular, por isso utiliza-se o termo NoSQL. De acordo com Lopes (2010), apesar do ganho no desempenho a transição é considerada como complexa. Atualmente o banco de dados Cassandra é baseado na tecnologia emergente *NoSQL* e encontra-se mantido pela Fundação Apache [21].

Vale ressaltar que o *Big Data* do Google foi uns dos motivos para a ascensão dos bancos NoSQL, mas como cita [22], ele não foi o único motivo. Uns dos motivos para a escolha desse software, é a facilidade do uso de bancos NoSQL em ambientes distribuídos. Desse modo, o Cassandra, é um projeto de alto nível da fundação Apache

iniciado pelo Facebook e montado tendo como base os aspectos estruturais do *Dynamo* e *BigData*. Suas características são a escalabilidade linear, tolerância a falha de forma transparente, alta disponibilidade, alta performance, particionamento de dados e gerenciamento de dados estruturados, semiestruturados e não estruturados [20].

2. NoSQL

O termo NoSQL foi primeiramente utilizado em 1998 para um banco de dados relacional que omitiu o uso de SQL [9]. O NoSQL (*Not Only SQL*) inclui todos os tipos de bancos de dados não relacionais, e foi criado com o objetivo de atender aos requisitos de gerenciamento de grandes volumes de dados, semiestruturados ou não estruturados, que precisam de alta disponibilidade e escalabilidade.

O Google e a Amazon foram uma das primeiras empresas necessárias para armazenar grandes quantidades de dados. Eles essencialmente descobriram que armazenar dados em um banco de dados relacional não permitia que eles armazenassem a grande quantidade de dados de maneira econômica. Eles buscaram abordagens alternativas com sucesso e publicaram suas descobertas em documentos seminais**. Bigtable: Um sistema de armazenamento distribuído para dados estruturados** e **Dynamo: o armazenamento de valor-chave altamente disponível da Amazon** respectivamente [5].

Bancos de dados NoSQL possibilitam uma escalabilidade menos complicada e mais acessível, pois não há necessidade de máquinas muito poderosas e sua facilidade de manutenção possibilita um número menor de profissionais [9]. Com isso, eles vão ficando mais conhecidos entre as grandes empresas, devido à sua capacidade de dimensionar horizontalmente e de poder trabalhar com dados não estruturados e semiestruturados de maneira eficiente.

3. Modelo de Banco de Dados NoSQL

Os tipos de modelos NoSQL podem ser agrupados em 4 tipos bem distintos. São eles: Chave-Valor (*Key-Value*), Orientado a Documentos, Orientado a Colunas e Orientado a Grafos.

3.1 Banco de Dados Chave-Valor (*Key-Value*)

O modelo chave-valor é bastante simples e nos possibilita a visualização do banco de dados através de uma tabela HASH, na qual todo o banco de dados é composto por um

conjunto de chaves, onde elas estão relacionadas a um único valor. Na **Figura 1** mostra um exemplo de como é armazenada as informações em um banco de dados chave-valor, nessa figura pode-se observar os campos e suas instâncias [9].

Este modelo de banco é de fácil implementação, permitindo que os dados sejam acessados rapidamente pela chave, o que contribui para o aumento da disponibilidade do acesso aos dados. Para a manipulação dos dados, é utilizado os comandos simples de get() (Retornar o valor) e o set() (Capturar o valor). Um problema desse tipo de banco de dados é que ele não permite a recuperação de objetos por meio de consultas mais complexas. Um bom exemplo é o banco de dados Dynamo.

Figura 1. Modelo Chave-Valor

Key	Value
Name	Joe Bloggs
Age	42
Occupation	Stunt Double
Height	175cm
Weight	77kg

Fonte:Internet

3.2 Banco de Dados Orientado a Documentos

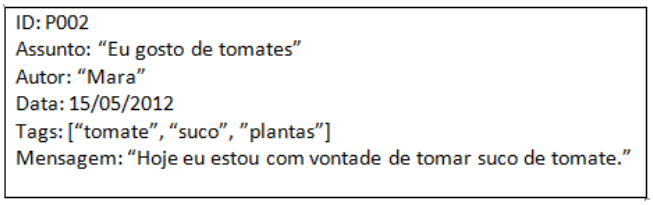
Como o nome já diz, este modelo de banco armazena coleções de documentos. Um documento é um objeto com um identificador único e um conjunto de campos que podem ser strings, lista ou documentos aninhados.

No modelo orientado a documentos, temos um agrupamento de documentos e em cada um destes documentos há um conjunto de campos e o valor deste campo, diferente do modelo chave-valor, onde uma única tabela hash é criada para todo o banco de dados. Outra característica importante sobre o modelo orientado a documentos é a ausência de um esquema rígido, ou seja, não há necessidade de uma estrutura fixa. Com isso, é possível ocorrer atualizações no documento, sem causar problemas no mesmo.

Na **Figura 2** temos o exemplo de um determinado documento que foi definido por: *Assunto, Autor, Data, Tags e Mensagens*, caso adicione um novo campo não haverá nenhum problema, pois uma das principais vantagens desse modelo é essa facilidade e

flexibilidade em atualizar a estrutura dos documentos, sem prejudicar o banco de dados [9].

Figura 2 – Modelo Orientado a Documento

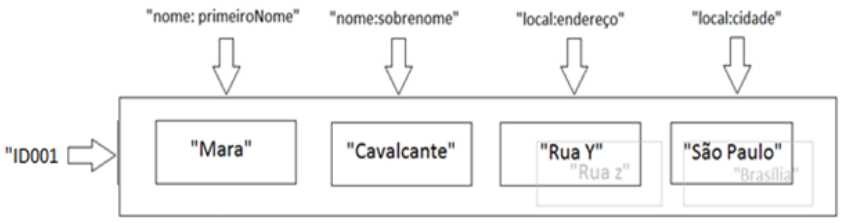


Fonte: Internet

3.3 Banco de Dados Orientado a Colunas

O modelo orientado a colunas é um pouco mais complexo que o de chave-valor. Este tipo de modelo os dados são indexados por uma tripla (coluna, linha e timestamp), onde as linhas e colunas são identificadas por chaves e o timestamp é o que permite diferenciar as versões de um mesmo dado [9]. As operações de leitura e escrita deste modelo são atômicas, ou seja, todos os valores que estão relacionados a uma linha são considerados na execução da operação de leitura/escrita. Como o nome sugere, este modelo tem um conceito importante é o de família de colunas (*column family*), e tem como objetivo agrupar colunas que armazenam o mesmo tipo de dados. O banco orientado a colunas permite particionamento de dados, oferecendo forte consistência e não garante alta disponibilidade. Na **Figura 3** pode-se observar um exemplo do modelo orientado a colunas.

Figura 3 – Modelo Orientado a Coluna



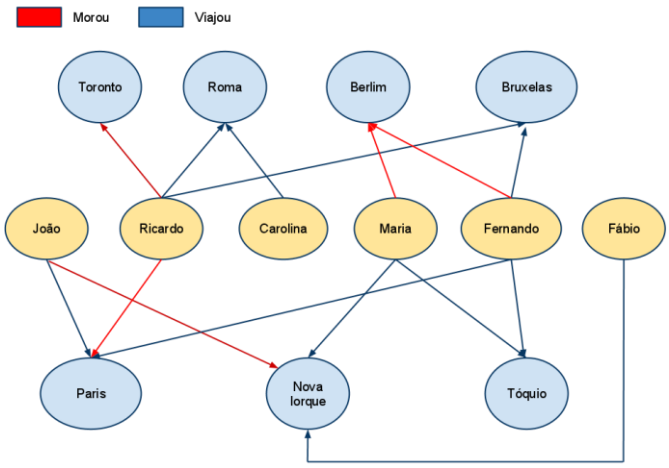
Fonte: Internet

3.4 Banco de Dados Orientado a Grafos

O banco orientado a grafos possui três componentes básicos: os nós que são os vértices, os relacionamentos que são as arestas e as propriedades também conhecidos como atributos dos nós [9]. Neste modelo, o banco de dados é visto como multigrafo rotulado e direcionado, onde cada nó pode ser conectado por mais de uma aresta.

A utilização deste modelo é muito vantajosa quando consultas complexas são determinadas pelos usuários. Esse banco de dados orientados a grafos tem uma alta performance, tendo assim um bom desempenho nas aplicações, diferente do modelo conceitual onde essas consultas se mostram mais complexas, pois não possibilitam que as informações sejam exibidas de uma forma natural.

Figura 4 – Modelo Orientado a Grafos



Fonte: Internet

4. BASE X ACID

Em geral a internet está criando uma quantidade enorme de dados, com esse constante crescimento os dados precisam ser processados, analisados e entregues aos usuários que requisitaram. Empresas, organizações e indivíduos que oferecem serviços nesse campo têm que determinar suas necessidades individuais em relação ao desempenho, confiabilidade, disponibilidade, consistência e durabilidade [15]. Com o número crescente de aplicações, sendo elas a maioria web, principalmente as de grande escala, a disponibilidade e tolerância a partição são as mais importantes do que a consistência. Essas aplicações devem ser confiáveis, o que causa disponibilidade e redundância. Usando a propriedade ACID torna-se difícil de se alcançar essas propriedades, desta maneira abordagens como BASE são utilizadas.

A abordagem BASE perde as propriedades ACID de consistência e isolamento a favor de ganhar “disponibilidade e desempenho”. Para entender melhor sigla ACID é composta das seguintes características:

- **A – Atomicidade:** A propriedade de atomicidade garante que as transações sejam atômicas (indivisíveis). A transação será executada totalmente ou não será executada [13].
- **C – Consistência:** A propriedade de consistência garante que o banco de dados passará de uma forma consistente para outra forma consistente [13].
- **I – Isolamento:** A propriedade de isolamento garante que a transação não será interferida por nenhuma outra transação concorrente [13].
- **D – Durabilidade:** A propriedade de durabilidade garante que o que foi salvo, não será mais perdido [13].

Como dito anteriormente, as aplicações web lidam com uma grande quantidades de dados, com isso elas não conseguem manter todas as propriedades ACID e uma boa performance das aplicações ao mesmo tempo, assim as empresas optaram por perder umas das propriedades para suprir suas necessidades mais importantes, e então surge as propriedades BASE que são:

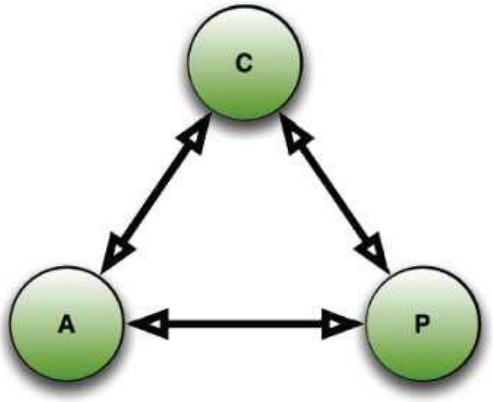
- **BA – (Basically Available) Basicamente Disponivel –** Prioridade na disponibilidade dos dados.
- **S - (Soft-State) Estado leve –** O sistema não precisa ser consistente o tempo todo.
- **E – (Eventually Consistent) Eventualmente Consistente** - Consistente em momento indeterminado.

Resume as propriedades BASE da seguinte maneira: uma aplicação funciona basicamente todo tempo (Basicamente Disponível) e não precisa ser consistente o tempo todo (Eventualmente Consistente), com isso a BASE se foca mais em disponibilidade e desempenho, que são as necessidades mais básicas de uma aplicação web, conceder os dados requisitado pelo usuário e fazer isso de uma forma rápida e simples.

5. Teorema CAP

O teorema CAP foi criado pelo Dr. Erick Brewer e foi apresentado pela primeira vez no simpósio da ACM, aonde ele apresentou o comportamento de um sistema distribuído. Este procedimento, explica um conjunto de nós interconectados que distribuem todos os dados, ou seja, o teorema CAP diz respeito sobre o comportamento de um sistema distribuídos quando se há necessidade de requisitar um comando de escrita, acompanhado de uma requisição de leitura de dados.

Figura 5 – Ilustrando a interação do teorema CAP



Fonte: Internet

- **C**onsistency (Consistência): Um cliente pode ler o dado no mesmo nó que este dado foi escrito ou de um nó diferente, o mesmo será retornado para a aplicação cliente. A consistência descreve como e se o estado de um sistema fica consistente após uma operação.
- **A**vailability (Disponibilidade): O sistema é tolerante a falhas de software ou hardware e geralmente também continua disponível durante atualizações de software e hardware. Com isso disponibilidade refere-se à compreensão e implementação de um sistema de modo que seja garantido que este permaneça ativo durante um determinado período de tempo.
- **P**artition Tolerance (Tolerância ao Particionamento): Esse comportamento refere-se à capacidade de continuar operando, mesmo que aconteça alguma falha na rede (quebra de conectividade), ou seja, isso garante que operações serão sempre completadas, mesmo que componentes individuais não estejam disponíveis.

O teorema CAP afirma que em um sistema distribuídos pode trabalhar com apenas duas dessas propriedades, mas nunca com os três ao mesmo tempo. Seguindo essa ideia podemos ter três tipos de sistemas:

- **Sistemas CA:** Sistemas que utilizam esse tipo de abordagem, com consistência forte e alta disponibilidade apresenta um risco para aplicação, pois não sabe lidar com falhas de uma partição. Caso ocorra essa falha, o sistema fica indisponível até que os membros do cluster se reestabeleçam novamente [11].
- **Sistemas CP:** Sistemas que precisam de consistência forte e tolerância a particionamento é necessário abrir mão da disponibilidade. Em sistemas do tipo CP, caso haja um particionamento e o sistema não entre em consenso, a escrita pode ser negada/rejeitada. Existem sistemas que trabalham para que nada de errado aconteça, tanto que não costuma existir, por exemplo, uma transação distribuída e sim um protocolo de consensos para garantir a forte consistência [11].
- **Sistemas AP:** Este tipo de sistema requer alta disponibilidade e tolerância a particionamento, e precisa abrir mão da consistência e aplicar o conceito de Consistência Eventual (Eventual-Consistency). Sistemas AP não podem ficar offline, a intenção é que esse tipo de sistemas permita mais escritas e tenham seus dados sincronizados em algum momento depois [11]. Então pode haver uma janela de inconsistência.

6. Cassandra

O Cassandra é um sistema de banco de dados baseado na abordagem NoSQL. Esse sistema foi desenvolvido pelo Facebook para resolver o problema com as suas buscas de sua caixa de entrada de mensagens. No ano de 2008 ele acabou se tornando open source e no ano seguinte ele passou a ser mantido pela fundação Apache. O Cassandra é baseado no tipo chave/valor, nesse tipo de banco os dados são apresentados através de uma chave.

O Cassandra é considerado um banco AP (Availability / Partition Tolerance) por fornecer um sistema de armazenamento distribuído, altamente escalável e eventualmente consistente. Para garantir isso foram unidas características de dois sistemas NoSQL, o Dynamo da Amazon [18] e BigTable do Google [19].

O Dynamo surgiu a partir do desejo de se ter um banco de dados simples, altamente escalável, confiável para lidar com grandes demandas de leitura e escrita.

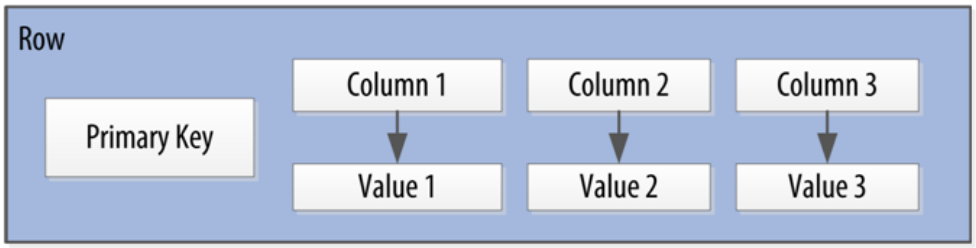
Durante a Black Friday de 2004 nos Estados Unidos, a Amazon sofreu com uma série de momentos de indisponibilidade do seu site, causando sobrecargas em boa parte do banco relacional utilizado na época, ocasionando um prejuízo financeiro para o site de e-commerce. Com isso foi criado e utilizado o banco de dados Dynamo para tratar os sistemas que sofriam com uma demanda maior. A Amazon disponibilizou um documento explicando como funcionava a arquitetura Dynamo, esse documento serviu como base para a criação de outros bancos de dados NoSQL.

No ano de 2004 o BigTable começou a ser desenvolvido pelo Google, também como uma solução altamente escalável e distribuída. Mas tinha como objetivo armazenar um grande volume de dados de indexação de todas as páginas web mapeadas pelo Google que então era usado para alimentar o seu motor de buscas.

6.1 Estrutura dos dados no Cassandra

Como mencionado antes, o Cassandra é um banco do tipo NoSQL baseado no modelo chave-valor, na qual cada grupo de chave-valor é semelhante ao que seria uma coluna, e um conjunto de colunas relacionadas a uma chave primária (*primary key*) integram o que seria uma linha de uma tabela no Cassandra, como mostrado na **Figura 6.**

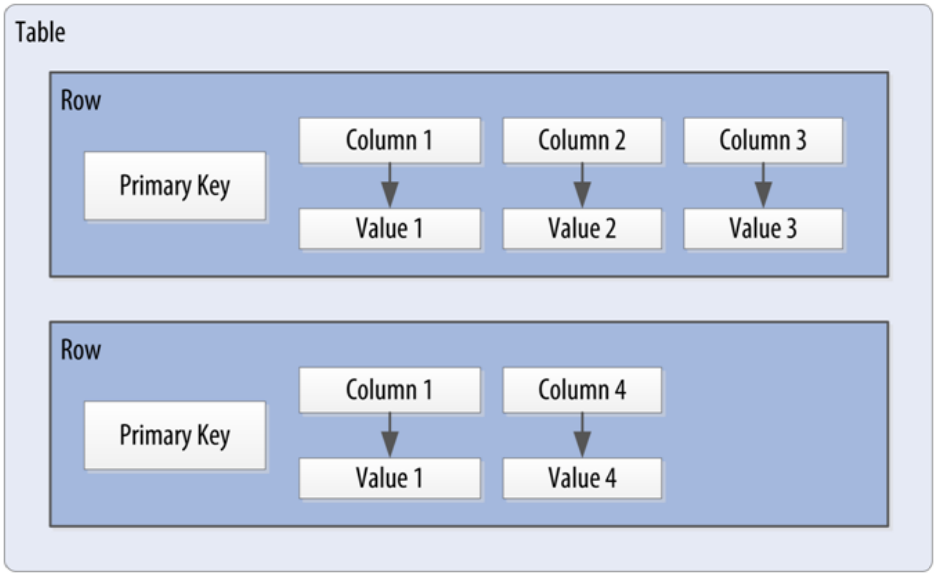
Figura 6 – Estrutura de uma linha de uma tabela no Cassandra



Fonte: Internet

A proposta de tabelas no Cassandra é muito semelhante às tabelas encontradas em bancos relacionais, elas têm como finalidade de agrupar dados equivalentes à uma determinada entidade [7]. No banco relacional uma coluna sempre terá que ser preenchida com algum valor, mesmo sendo um valor nulo, diferente do Cassandra aonde não há necessidade de popular as colunas ao gerar uma nova linha, de modo que poderá ter larguras distintas para cada uma delas. A **Figura 7** demonstra esse conceito.

Figura 7 – Exemplo de uma tabela do Cassandra



Fonte: Internet

Um *timestamp* é sempre agregado quando se grava um valor em uma coluna, para informar o horário exato em que esse valor foi gravado [7]. Isto é interessante, pois sempre irá identificar versões desatualizadas de valores quando o Cassandra equiparar o resultado de uma consulta feita em vários nós diferentes.

Fonte: Internet

6.2 Arquitetura Apache Cassandra

Como foi mostrado, o teorema CAP [14] pode atender plenamente apenas dois itens entre consistência, disponibilidade e tolerância, com isso o Cassandra contribui grandemente para que ele seja capaz de escalar, executar e oferecer disponibilidade. O Cassandra possui uma arquitetura muito robusta e complexa, para garantir que um grande volume de dados seja tratado de forma rápida e eficiente. A arquitetura do Cassandra é composta por diversos sistemas distribuídos que asseguram que cada parte do sistema funcione corretamente.

Caso uma requisição de leitura ou escrita de dados seja feita, qualquer nó do cluster do Cassandra pode trata-la. Através do uso da chave, o nó que recebeu a requisição consegue saber quais são os nós que apresentam informações dos dados. Com isso, o sistema irá guardar até que um número configurado de réplicas responder com o dado (leitura) ou com uma confirmação (escrita).

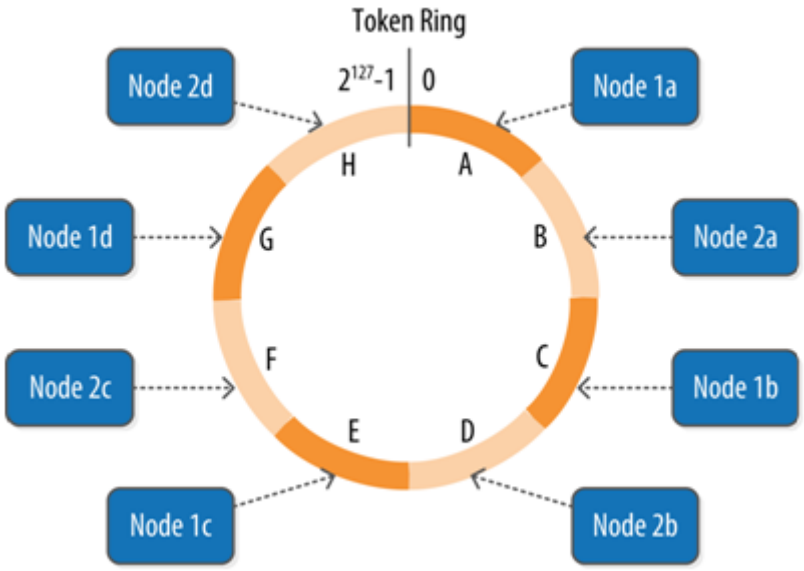
No Cassandra o cluster é organizado em um anel, aonde cada nó do anel possui um intervalo de valores determinados. Esses dados são separados entre vários nós através

de um *hashing* da chave identificadora de dados. Com isso, esse hash gera um valor que determina, através do intervalo de cada nó do anel, quais das maquinas é responsável por determinado dado. As máquinas responsáveis são chamadas de coordenadoras.

Com essa divisão dos nós em anel simplifica a adição e remoção de nós no cluster. Caso um nó seja acrescentado ou excluído no cluster, somente os nós vizinhos serão afetados. Isso contribui para que o sistema seja bastante escalável, já que a adição de um nó no cluster não afeta o funcionamento de todo o sistema.

No Cassandra a replicação é utilizada para garantir a alta disponibilidade de dados. Ao contrário de esquemas de replicação complicados em vários bancos de dados tradicionais ou outros bancos de dados NoSQL, a replicação no Cassandra é extremamente fácil de configurar. O responsável por replicar os dados entre N-1 nós, é o coordenador do nó, que foi definido anteriormente. Existem três formas diferentes de fazer essa replicação, são elas: Rack Unaware, Rack Aware e Datacenter Aware. No modo Rack Unaware, o coordenador simplesmente seleciona os próximos N-1 nos do anel. Nos modos Rack Aware e Datacenter Aware, e utilizado um sistema externo que elege um líder, que tem como função avisar cada no que se conecta ao cluster a faixa de valores do anel que ele será uma réplica. A **Figura 9** tem uma demonstração da arquitetura em anel.

Figura 9 – Representação da arquitetura em anel



Fonte: Internet

7. Conclusão

Bancos de dados NoSQL podem ser uma alternativa real para bancos que necessitam de uma alta escalabilidade, desempenho e que podem possuir tolerância a falhas, o NoSQL tem uma desvantagem que nem sempre é possível garantir a consistência dos dados, mas ele não deixa de ser uma ótima opção para solução de cargas elevadas de dados.

O Cassandra é uma boa solução de alta disponibilidade e escalabilidade quando há necessidade de ser trabalhar com sistemas que demandam volumes de leitura e escrita extremamente alto e em que sempre haverá a chave-primária em mãos. Devido a sua arquitetura ser derivada do Dynamo o Cassandra é bom para soluções em nuvens, permitindo uma alta elasticidade, desta forma, ele pode se adequar a complexidade da infraestrutura de acordo com a demanda de leitura/escrita necessária.

Por outro lado, o Cassandra não é interessante para ser utilizados em cenários com uma baixa demanda de leitura/escrita, e também não é recomendado para ser utilizado como banco de dados analítico, por haver várias limitações quando não são feitas consultas utilizando a chave-primária.

REFERÊNCIAS DE FIGURAS

Figura 1 – Disponível em <https://www.guru99.com/nosql-tutorial.html> Acesso em Jun 2019.

Figura 2 - Disponível em < https://www.devmedia.com.br/banco-de-dados-nosql-um-novo-paradigma-revista-sql-magazine-102/25918> Acesso em Jun 2019.

Figura 3 – Disponível em < https://www.devmedia.com.br/banco-de-dados-nosql-um-novo-paradigma-revista-sql-magazine-102/25918> Acesso em Jun 2019.]

Figura 4 – Disponível em < https://www.devmedia.com.br/banco-de-dados-nosql-um-novo-paradigma-revista-sql-magazine-102/25918> Acesso em Jun 2019.

Figura 5 - Disponível em < https://www.infoq.com/articles/graph-nosql-neo4j/> Acesso em Jun 2019.

Figura 6 - Disponível em < https://busy.org/@billigeplaetze/adeepdiveintoapachecassandra-part1datastructure-0lobs1yppr> Acesso em Jun 2019.

Figura 7 - Disponível em < https://busy.org/@billigeplaetze/adeepdiveintoapachecassandra-part1datastructure-0lobs1yppr > Acesso em Jun 2019.

Figura 8 - Disponível em <https://www.devmedia.com.br/introducao-ao-cassandra/38377 > Acesso em Jun 2019.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] Disponível em: < http://www.cs.berkeley.edu/~brewer/cs262b-2004/PODC-keynote.pdf > Acesso em: Junho de 2019.

[2] Disponível em: < https://unrealps.wordpress.com/2010/12/28/o-teorema-cap/ > Acesso em: Junho de 2019.

[3] Disponível em: < https://medium.com/@jrobertoaraujo/teorema-cap-3094645d7249 > Acesso em: Junho de 2019.

[4] Disponível em: < https://medium.com/nstech/apache-cassandra-8250e9f30942 > Acesso em: Junho de 2019.

[5] Disponível em: < https://dzone.com/articles/introduction-nosql-apache > Acesso em: Junho de 2019.

[6] Disponível em: < https://dzone.com/articles/introduction-apache-cassandras > Acesso em: Junho de 2019.

[7] Disponível em: < https://www.devmedia.com.br/introducao-ao-cassandra/38377 > Acesso em: Junho de 2019.

[8] Disponível em: < https://www.devmedia.com.br/banco-de-dados-nosql-um-novo-paradigma-revista-sql-magazine-102/25918 > Acesso em: Junho de 2019.

[9] Disponível em: < https://www.devmedia.com.br/introducao-aos-bancos-de-dados-nosql/26044 > Acesso em: Junho de 2019.

[10] Disponível em: < https://pt.slideshare.net/Celio12/trabalho-no-sql-aricelio-de-souza > Acesso em: Junho de 2019.

[11] Disponível em: < https://blog.caelum.com.br/nosql-do-teorema-cap-para-paccl/ > Acesso em: Junho de 2019.

[12] Disponível em: < http://blog.nahurst.com/visual-guide-to-nosql-systems > Acesso em: Junho de 2019.

[13] Disponível em: < http://blog.lucasrenan.com/propriedades-acid/ > > Acesso em: Junho de 2019.

[14] J.Browne, “Brewer's CAP Theorem”, 2009.

Available:http://www.julianbrowne.com/article/viewer/brewers-captheorem

[15] Gray, Jonathan: *CAP Theorem*. August 2009. – Blog post of 2009-08-24. http://devblog.streamy.com/2009/08/24/cap-theorem/

[16] Vogels, Werner: *Eventually consistent*. December 2007. – Blog post of 2007-12-19. http://www.allthingsdistributed.com/2007/12/eventually_consistent.html

[17] Brewer, Eric A.: *Towards Robust Distributed Systems*. Portland, Oregon, July 2000. – Keynote at the ACM Symposium on Principles of Distributed Computing (PODC) on 2000-07-19.

[18] 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

[19] 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

[20] Datastax. **A Brief Introduction to Apache Cassandra** (2015). Disponível em: <https://academy.datastax.com/resources/brief-introduction-apache-cassandra > acesso em: abril de 2016.

[21] Faria, Alessandro Oliveira. **Apache Cassandra NoSQL, uma tecnologia emergente**. Disponível em: <http://www.linhadecodigo.com.br/artigo/2840/apache-cassandra-nosql-uma tecnologia-emergente.aspx> Acesso em: abril de 2016.

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

[23] Pereira, Srinath **Considerações sobre o Banco de Dados Apache Cassandra** (2012). Disponível em: https://www.ibm.com/developerworks/br/library/os-apache-cassandra. Acessado em: abril de 2016.

[24] Soares. Ricardo Rocha. **Estudo comparativo entre sistemas de bancos de dados de Grafos e relacionais para a gerência de dados de proveniência em workflows científicos**. Dissertação, COPPE – UFRJ (2013).

[25] Sousa, Paulo. **O Teorema CAP (2010)**. Disponível em: <https://unrealps.wordpress.com/2010/12/28/o-teorema-cap/> acesso em: abril de 2016.