Banco de dados de chave-valor distribuído YEK

Silvana Trindade, Maurício André Cinelli

13 de agosto de 2015

Descrição

Existem vários modelos de armazenamento de dados noSQL (*Not Only SQL*), como por exemplo: chave-valor, documento, colunas, e grafos. O modelo chave-valor, apresenta a proposta que permite que o desenvolvedor efetue a persistência de dados totalmente livre de definições de estrutura para esquema [4], este trabalho terá foco neste modelo. No modelo orientado a grafo, o banco de dados é visto como um multigrafo rotulado e direcionado, onde cada par de nós pode ser conectado por mais de uma aresta.

Em sua essência o orientado a documentos é semelhante ao chave-valor [4]. Entretanto em vez de armazenar qualquer arquivo binário como valor de uma chave é requisitado que o valor dos dados que serão armazenados possua um formato que o banco possa interpretar. No modelo orientado a colunas os dados são indexados por uma tripla (linha, coluna e timestamp), onde linhas e colunas são identificadas por chaves e o timestamp permite diferenciar múltiplas versões de um mesmo dado [5].

Quando se trata de alta performance e disponibilidade em softwares, geralmente utiliza-se apenas acesso em dados que sejam armazenados por chave-valor, ou seja, armazenam valores quaisquer indexados por chaves únicas. Isto torna possível grande escalabilidade horizontal, pois como não é um banco de dados relacional, não há ligação entre os dados, tornando-os independentes, podendo assim serem facilmente divididos em servidores dentro de um *cluster*. Em sistemas relacionais tradicionais, tratar escalabilidade de leitura e escrita é complexo, se não impossível de atingir, devido à forma que os dados são armazenados – um problema que em sistemas noSQL não ocorre.

O sistema chave-valor pode ser visto como uma tabela *hash* distribuída em grande escala, persistente e tolerante à falhas [1]. É importante destacar que este modelo de armazenamento não é suficiente e recomendado para todos os problemas de armazenamento, possuindo seus prós e contras: não é possível fazer consultas com filtros avançados; não possui chave estrangeira; *joins* precisam ser feitos em código.

Este modelo, porém, torna a performance previsível; é facilmente distribuída entre servidores; ajuda a separar dados e lógica; pode ser usado como mecanismo de cache, e muitos dos conceitos aqui geralmente são empregados em software quando a performance se torna necessária.

Uma característica de sistemas noSQL é que estes não garantem as propriedades conhecidas por ACID (A - atomicidade, C - consistência, I - isolamento, D - durabilidade), porém introduzem outro problema, conhecido por CAP (C - consistência, A - disponibilidade e P - tolerância à partição). Sistemas chave-valor tendem a ser *AP* ou *CP*.

O sistema a ser desenvolvido seguirá o modelo *AP*, no qual o foco está na tolerância à partição e na disponibilidade, com eventual consistência do banco de dados. O banco suportará somente chaves e valores em formato de string. Além disso, as consultas serão simples, sem utilizar join por exemplo.

Justificativa

O grande volume de dados gerado por aplicações Web, juntamente com os requisitos diferenciados destas aplicações, como a escalabilidade sob demanda e o elevado grau de disponibilidade, têm contribuído para o surgimento de novos paradigmas e tecnologias [5]. Dentro deste cenário surgiu o modelo noSQL. As principais características deste modelo são: ausência de esquema ou esquema flexível, suporte nativo a replicação, API simples para acesso aos dados e consistência eventual [5].

Banco de dados noSQL fazem o tratamento de um conjunto limitado de funções, como pode ser observado na figura 1. Além disso, o modelo chave-valor é considerado de fácil implementação, permitindo que os dados sejam rapidamente acessados pela chave, principalmente em sistemas que possuem alta escalabilidade, contribuindo também para aumentar a disponibilidade de acesso aos dados [5].

O fato de cada sistema tratar diferentes problemas e ambientes, faz com que haja espaço para que outros sistemas especializados sejam criados e ainda assim, sejam relevantes.

Banco de dados comerciais que utilizam do método chave-valor restringem o tratamento de dados, por exemplo o Voldemort suporta json e string entre outras; DynamoDB suporta número, string, booleano, binário e conjuntos destes tipos; e o BerkeleyDB suporta um conjunto extenso de tipos de dados. O sistema proposto neste trabalho se diferencia por tratar apenas chaves e valores string, que aumenta a performance do sistema, eliminando testes e conversões desnecessárias, além de simplificar a implementação.

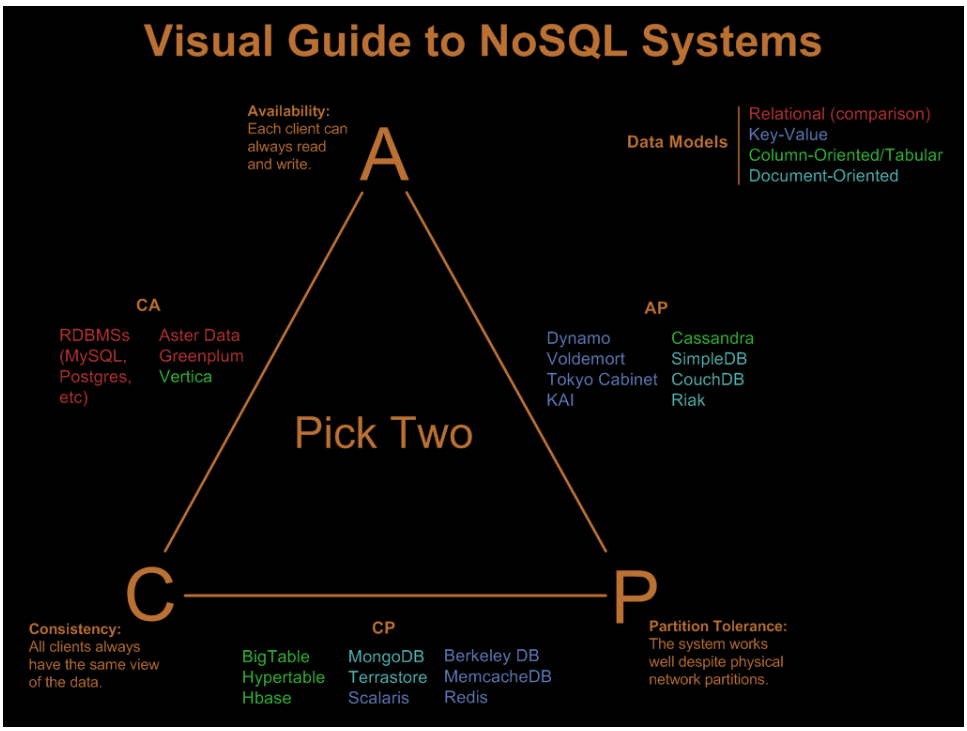

Figura 1: Guia visual do teorema CAP em sistemas NoSQL. Neste teorema, os sistemas tratam apenas dois dos três problemas. Em azul, estão representados os sistemas de chave-valor. Observe que diferentes sistemas possuem abordagens diferentes no que decidem como prioridade. Fonte [2].

Para manter a consistência dos dados entre os servidores, utiliza-se de árvores hash ou árvores Merkle. Estas árvores caraterizam-se por ter hashes dos dados armazenados nos nós folha, e os nós pais contêm hashes dos seus filhos [3]. As árvores binárias perfeitas todos os nós folhas possuem a mesma profundidade ou estão no mesmo nível. Uma árvore binária perfeita com profundidade h contêm 2^h folhas, em outras palavras, se um intervalo contêm n tokens então a árvore Merkle que representa ele irá conter $log(n)$ níveis [6].

Este mecanismo permite verificar de forma mais eficiente quais dados estão com falhas ou faltando em diferentes servidores, visto que somente os dados que precisam ser alterados serão enviados pela rede [6].

Tabela 1: Cronograma

Atividade	11-18/08	18-25	25-01	01-08	08-15	15-22	22-29	29-06	06-13	13-20	20-27	27-03	03-10	10-17	17-24/11
Estudar banco de dados noSQL	X														
Estudar outros bancos chave-valor	X														
Estudar técnicas de armazenamento distribuído	X														
Definir a estrutura		X													
Implementar instanciação do cluster		X													
Implementar armazenamento e distribuição			X	X	X										
Estudar e implementar Merkle tree					X	X									
Testes para Merkle tree							X								
Implementar inserção								X							
Implementar busca									X						
Tratamento de falhas										X	X				
Implementar atualização												X	X		
Implementar exclusão													X		
Correção de bugs			X	X	X	X	X	X	X	X	X	X	X	X	X

Tabela 2: Para cada etapa de implementação, será realizado baterias de testes para garantir a funcionalidade adequada do software. Repositório em https://github.com/Trindad/yek

Referências

[1] Project-Voldemort - A distribuited database, http://www.project-voldemort.com/voldemort/

[2] Visual Guide to NoSQL Systems, http://blog.nahurst.com/visual-guide-to-nosql-systems

[3] Merkle Signature Schemes, Merkle Trees and Their Cryptanalysis. BECKER, Georg. http://bit.ly/1J1mgB9

[4] Abordagem NoSQL – uma real alternativa. TOTH, Renato Molina http://bit.ly/1MhTflu

[5] NoSQL no desenvolvimento de aplicações Web colaborativas. LÓSCIO, B. F.; OLIVEIRA, H. R. de e PONTES, J. C. de S. http://www.addlabs.uff.br/sbsc_site/SBSC2011_NoSQL.pdf

[6] Using Merkle trees to detect inconsistencies in data http://distributeddatastore.blogspot.com.br/2013/07/cassandra-using-merkle-trees-to-detect.html