05

Introdução ao Elasticsearch

A complexidade da busca

Procurar por registros em meio a grandes volumes de dados é uma tarefa complicada e há muitos anos recebe atenção de diversas áreas de pesquisa. Por décadas, bancos de dados relacionais foram os principais motores por trás de qualquer solução de busca.

Diversos bancos de dados oferecem a capacidade de busca cheia em texto (*full-text search*). Habilitar este recurso necessita, em geral, de tarefas administrativas como instalação de *add-ons* trocas de configurações do banco e execuções de *procedures*.

Seria ideal uma solução mais simples, flexível e portável, afinal de contas, não queremos contratar um DBA (Administrador de Banco de Dados) ou nos tornarmos especialistas em um banco de dados para poder fazer buscas que vão além do famoso e nada eficiente `campo like %valor%`.

Uma outra dificuldade em lidar com buscas está ligada à maneira com que a busca em si é orientada. Tanto em bancos de dados relacionais, quanto em bancos de dados não relacionais, conhecidos como *NoSQL*, buscas são orientadas a colunas ou atributos chaves que devem ser indexados previamente.

Não queremos também estar atrelados a campos específicos e exigir que nosso usuário precise saber detalhes do nosso sistema para poder extrair a informação que ele necessita. Por exemplo, imagine que queremos encontrar em uma tabela todas as vendas de computadores que ocorreram no estado de São Paulo na última semana. Para termos uma busca eficiente, precisamos que boa parte destes campos estejam previamente indexados, caso contrário a busca pode acabar varrendo a tabela toda.

Além disso, se liberamos esta busca para usuários de um sistema, somos praticamente obrigados a colocar um formulário de busca. Imagine o "sucesso" (sarcasmo) que o Google teria caso oferecesse um formulário de busca, em vez de um campo livre de busca.

Engine de busca

Felizmente, existem bibliotecas que nos ajudam a ter bastante flexibilidade em nossas buscas. Um exemplo é o Apache Lucene (https://lucene.apache.org/core), que é uma biblioteca em Java que oferece um *motor de busca (search engine)* poderoso.

Lucene é orientado a documentos e possui funcionalidades como busca exata, busca por similaridade, *highlight* do termo procurado no resultado, busca por termos em documento e busca por frase, além de também suportar analisadores de texto em diversos idiomas. Ainda que o Lucene seja excelente do ponto de vista de *search engine*, ele possui alguns aspectos que limitam sua adoção. Por exemplo, como o Lucene é escrito em Java, é difícil utilizá-lo diretamente com linguagens de programação como Ruby ou C#.

O Lucene foi desenhado para ser executado em uma única JVM. Isso nos limita tanto no volume de dados, afinal toda informação fica armazenada em um único nó, não temos tolerância a falhas e alta disponibilidade, afinal, caso este nó tenha problemas, nosso *search-engine* estaria indisponível.

Dados os volumes de dados e as necessidades de disponibilidade atuais, ter um sistema que depende de uma única máquina (seja virtual ou física) é de longe um grande problema. Estamos interessados no poder de busca oferecido pelo

Lucene, porém em um ambiente na nuvem, com alta disponibilidade e escalabilidade horizontal, afinal queremos poder adicionar mais máquinas (baratas) quando for necessário.

ElasticSearch: Lucene e análise de dados na Nuvem

ElasticSearch (https://www.elastic.co/products/elasticsearch) é uma implementação de código aberto disponibilizada sob a licença Apache 2, suportada pela empresa Elastic.co que traz o poder do Lucene para o ambiente da nuvem. O ElasticSearch aborda os problemas descritos acima, tais como distribuição de dados e alta disponibilidade, tomando conta de detalhes como replicação de dados e tolerância a falhas de hardware. Com o aumento do poder de processamento que a nuvem nos dá, o ElasticSearch suporta também operações como agregações e buscas geo-espaciais que escalam horizontalmente ao nível de terabytes de dados.

Vale destacar que o ElasticSearch tem sido muito utilizado como ferramenta de análise de dados (*data analytics*), já que ele nos oferece a combinação de um poderoso *full-text search engine* com *data analytics* em larga escala. Como um exemplo real, imagine que queremos saber os 100 produtos mais vendidos no último ano e os seus valores médios de venda. Este problema é relativamente fácil de resolver em um banco de dados comum. Porém, queremos também limitar nossos resultados a produtos que tiveram bons comentários de clientes. Existem algumas frases (e não palavras) chaves com palavras que devemos utilizar na busca e nos ajuda a definir o que é um comentário positivo ou negativo.

A interação com o ElasticSearch, diferentemente da interação com o Lucene, acontece com o uso do formato JSON através de Restful APIs. Tal abordagem que nos permite ter um *search engine* em Java que interopera com diversas linguagens de programação. ElasticSearch oferece também como opção bibliotecas oficiais em diversas linguagens, como Javascript, Python, Java, Groovy, Perl, PHP, Ruby e .Net que abstraem os detalhes da interação via protocolo HTTP.