02

Melhorando o desempenho com Cache

Transcrição

Quando trabalhamos em um projeto real, a quantidade de usuários acessando a aplicação costuma ser um problema bem comum e que devemos discutir. Em tese, cada usuário acessando seus recursos no sistema geraria uma *query* no banco de dados, inclusive em momentos onde isso não é necessário, como em registros que demoram a atualizar. O problema é que o acesso ao banco de dados é uma operação computacionalmente cara. Ademais, lidar com muitos acessos em um único banco pode torná-lo indisponível por algum tempo. Essa situação induziu a pensar em estratégias para aumentar a escalabilidade com distribuição em cenários onde não são necessários.

Pensando em performance

Imagine que realizamos uma consulta para buscar os dados de um produto cadastrado no sistema. Uma vez que um produto é registrado, em raras ocasiões (como no caso de alguma promoção), ocorre sua edição. Porém, toda vez que buscamos os dados desse produto disparamos uma *query* contra o banco de dados que, muitas vezes, retornará o mesmo resultado.

Para melhorar a performance podemos colocar os resultados das buscas em algum lugar intermediário, na própria aplicação, para ser consultado antes de ir ao banco. Assim, para uma determinada busca, consultamos, primeiro, o *espaço intermediário* e, caso seja encontrado um para essa busca (*hit*), nós o utilizamos. Caso contrário, realizamos efetivamente, a consulta ao banco de dados e armazenamos o resultado no espaço. Costuma-se dizer que usar JPA torna a aplicação mais lenta por possuir uma camada sobre o JDBC. Entretanto, a própria especificação possui várias estratégias de otimização que antes eram inexistentes.

> Repare que, só faremos de fato uma comunicação remota ao banco de dados, caso não forem encontrados resultados guardados no *espaço intermediário* (*miss*). Esse *espaço* é o que comumente chamamos de *cache*.