Gerando Código Nativo: Vantagens e Desvantagens

por: Gleydson Lima
gleydson@jspbrasil.com.br

Em busca de um código mais rápido, alguns programadores acreditam que devem trocar a facilidade de transporte entre plataformas do Java pela velocidade bruta. Isso é feito compilando-se o código-fonte em Java para o código binário nativo de uma determinada plataforma, na qual o código é executado. Em vez de o fonte Java ser compilado para um formato de bytecode intermediário, ele é compilado diretamente para instruções de máquinas nativas.

A vantagem de se fazer isso é que você não tem mais o overhead envolvido na conversão do formato de bytecode intermediário para o código binário nativo em runtime. Isso geralmente resulta em um código com execução mais rápida. Sua desvantagem é que o java normalmente perde sua facilidade de mudança de plataforma, embora isso possa não ser um problema. Há pelo menos uma solução para esse problema, que não exige a perda da facilidade de transporte de plataforma, enquanto consegue os benefícios da execução do código binário nativo.

Primeiro, considere a geração completa de código estático, por onde você planeja abandonar a portabilidade da plataforma Java. Se o código tiver que rodar apenas em uma máquina, então a geração estática de código pode ser uma solução parcial para os problemas de desempenho. Um exemplo de uso dessa técnica é quando se escreve um código para uma aplicação de servidor. O código irá rodar em um servidor, e terá que ser muito rápido.

A geração de código estático também é útil quando se escreve código para um dispositivo embutido. O dispositivo pode ter requisitos de memória pequenos. A compilação do Java para o código binário nativo pode resolver esse problema, pois você se beneficiará com o tamanho reduzido do runtime necessário para o código.

Vários fornecedores possuem soluções nativas para o Java. Muitas dessas soluções possuem uma coisa em comum: elas tratam o Java simplesmente como uma linguagem de programação, e não como uma solução interplataforma e portátil, para o qual foi projetada. Você precisa considerar a inteireza da solução do fornecedor. Por exemplo, sua solução trabalha com carregamento dinâmico de classe? Existem outros recursos do Java, nos quais se você se baseia, que não são oferecidos pela solução?

Os programadores normalmente pensam que, se decidirem compilar o Java estaticamente, eliminarão a necessidade da JVM. Isso não é inteiramente verdade, dependendo da solução nativa que é usada. A compilação estática para o código binário nativo apresenta um problema com a coleta de lixo. Se todo o código for compilado para o código binário nativo, com a coleta de lixo irá funcionar sem uma JVM?

Vários fabricantes possuem diferentes soluções para esse problema. Uma é compilar o Java estaticamente, mas distribuí-lo no sistema de destino com uma pequena máquina virtual que realiza a coleta de lixo. Essa solução é vantajosa porque você não precisa considerar o gerenciamento de memória. No entanto, você não tem um programa tão pequeno quanto poderia precisar.

Uma segunda solução escolhida por alguns fornecedores é incluir uma palavra-chave delete em sua linguagem tipo Java. A desvantagem é que a inclusão de algo que não faça parte da *The Java Language Especification* torna imediatamente qualquer código que você escreve não portátil e incompatível entre as implementações do Java. A vantagem dessa técnica é que você elimina a necessidade de um runtime para realizar a coleta de lixo. E a desvantagem disso é que a coleta de lixo é realizada pelo programador que escreve o código, usando a palavra-chave especial *delete* .

Outra solução é considerar a compilação de partes do seu programa para o código binário nativo, mantendo a portabilidade de plataforma do Java. Com isso, você retém os benefícios do Java, como a coleta de lixo e o carregamento dinâmico de classes, enquanto, tendo selecionado os métodos importantes para o desempenho são compilados em código binário nativo. Isso é feito com ferramentas que compilam métodos Java em particular para o código nativo otimizado, que é então acessado por meio da JNI (Java Native Interface). Essa solução oferece o melhor dos dois mundos. Você mantém a portabilidade de plataforma do Java enquanto aproveita os benefícios de desempenho da execução de métodos nativos otimizados.

Considere cuidadosamente as vantagens e desvantagens da compilação para o código binário nativo. Depois de pesar os lucros e as perdas, você poderá tomar uma decisão consciente a respeito das escolhas.