JFLOAT: Uma Biblioteca de Ponto Flutuante com Arredondamento Direcionado para a Linguagem Java

José Frank Viana da Silva
Bacharelado de Sistemas de Informação , Universidade Potiguar,
Natal, RN
jfrank1500@gmail.com

Benjamín René Callejas Bedregal, Regivan Hugo Nunes Santiago
Departamento de Informática e Matemática Aplicada, UFRN
Natal, RN
bedregal@dimap.ufrn.br, regivan@dimap.ufrn.br

Resumo: *JFloat é uma implementação de software do padrão IEEE-754 de aritmética de ponto flutuante binária, a qual foi construída para prover algumas características não implementadas em Java, em especial o suporte ao arredondamento direcionado e flags de exceção. Apesar de programas escritos em Java, a princípio, serem portáveis para qualquer arquitetura ao usar operações de ponto flutuante, observou-se que programas que usam tipos de ponto flutuantes nativos de Java podem ser dependentes da máquina e do sistema operacional. JFloat se apresenta como uma possível solução para este problema.*

Palavras-chave: Java, biblioteca de software, arredondamento direcionado, portabilidade.

Introdução

Programas escritos em Java deveriam ser portáveis ao usar operações de ponto flutuantes, principalmente porque o padrão IEEE-754 especifica que programas deveriam ter precisamente o mesmo comportamento em toda configuração que segue o padrão [3].

Porém, programas que usam os tipos de ponto flutuantes nativos da linguagem Java podem ser dependentes do hardware e do sistema operacional sobre o qual executam [1].

Apesar da grande portabilidade de programas escritos em Java, existem alguns problemas com a linguagem, em relação ao seu suporte a aritmética de ponto flutuante que não segue fielmente o padrão IEEE-75 [4]:

- Embora IEEE-754 especifique cinco tipos de flags de exceção (operação inválida, overflow, divisão-por-zero, underflow e resultado inexato), a especificação de Java para ponto flutuante não usa flags de exceção. Por exemplo, em Java, um overflow resulta um infinito sinalizado, um underflow resulta em um zero sinalizado, e operações matematicamente indefinidas resultam em um Not a Number (NaN), porém nenhuma exceção é levantada.
- IEEE-754 estabelece que todo o hardware/software deveria fornecer ao programador um modo para especificar dinamicamente um dos quatro modos de arredondamento: arredondamento para o mais próximo (padrão), arredondamento para 0, arredondamento para cima e arredondamento para baixo. Porém, Java só suporta o modo de arredondamento para o mais próximo. Portanto, para aplicações que requeiram outros modos de arredondamento, o suporte nativo de Java para ponto flutuante não é adequado.

Biblioteca de ponto flutuante JFloat

Uma solução apresentada para os problemas apresentados acima foi a construção de uma biblioteca escrita em Java que implementa o padrão IEEE-754.

Em vez de usar os tipos de ponto flutuantes nativos da linguagem, a biblioteca JFloat usa uma representação interna e independente de máquina para executar operações de ponto flutuantes.

JFloat tem como características principais:

- **Compatibilidade entre versões de Java**. Compatível com a versão 1.4.2 e superiores de Java. Qualquer computador habilitado com Java pode usar a biblioteca de JFloat. A biblioteca foi criada usando o ambiente de desenvolvimento integrado Eclipse versão

3.1, mas qualquer compilador da linguagem Java pode ser usado para compilar o código fonte do projeto;

- **Independência de máquina**. JFloat representa números em ponto flutuante como variáveis inteiras. Esta característica assegura o mesmo comportamento em qualquer plataforma, porque são usadas somente operações sobre tipos inteiros para emular operações em ponto flutuantes;
- **Precisão**. A versão atual da biblioteca JFloat trabalha somente com números de ponto flutuantes em precisão simples armazenados em variáveis inteiras;
- **Operações**. Estão disponíveis as operações de adição, subtração, multiplicação, divisão, raiz quadrada, resto e comparação entre números;
- **Funções de conversão**. A biblioteca dispõe de funções de conversão para os formatos de ponto flutuante disponíveis na linguagem Java e números inteiros;
- **Disponíveis todos os modos de arredondamento**. Diferente do suporte nativo de Java a ponto flutuante e das bibliotecas de terceiros desenvolvidas para Java, JFloat oferece todos os modos de arredondamento requeridos pelo padrão IEEE-754;
- **Suporte a NaN e flags de exceção**. JFloat suporta o uso de QNaNs e SNaNs e após cada operação, os flags de exceção correspondentes são marcados de acordo com o resultado.

Descrição da API

Uma vez que Java, diferentemente de outras linguagens como C++, não possibilita a sobrecarga de operadores, as operações aritméticas elementares (adição, subtração, multiplicação, divisão, resto e raiz quadrada) foram definidas como funções (métodos de classe) que recebem operandos de ponto flutuante e retornam os resultados como definidos pelo padrão IEEE-754.

Todos os operadores aritméticos e funções auxiliares estão disponíveis como métodos públicos e estáticos da classe JFloat:

- float add(float x, float y): soma dois números de ponto flutuantes;
- float sub(float x, float y): subtrai dois números de ponto flutuantes;
- float mul(float x, float y): multiplica dois números de ponto flutuantes;
- float div(float x, float y): divide dois números de ponto flutuantes;
- float rem(float x, float y): retorna o resto da divisão de x por y, como definido no padrão IEEE-754;
- float sqrt(float x): retorna a raiz quadrada de x.

A tabela abaixo mostra um exemplo de código Java para somar dois números de ponto flutuante.

Tabela l: Exemplo de adição usando JFloat.

```
float a, b, c;
b = 0.5f;
c = 1.34f;
a = JFloat32.add(b,c);
```

Para identificar se o resultado de uma operação retornou um NaN, JFloat dispõe de duas funções isNaN e isSNaN. A primeira informa se o valor informado é um NaN. A segunda se é um Signaling NaN.

- boolean IsNaN(float x): retorna verdadeiro se x é um NaN e falso caso contrário;
- boolean IsSNaN(float x): retorna verdadeiro se x é um SNaN (Signaling NaN) e falso caso contrário.

As funções seguintes trabalham como operadores relacionais:

- boolean eq(float x, float y): retorna verdadeiro se x é igual a y, e falso caso contrário;
- boolean neq(float x, float y): retorna verdadeiro se x não é igual a y, e falso caso contrário;
- boolean eqSignaling(float x, float y): retorna verdadeiro se x é igual a y, e falso caso contrário. Uma exceção inválida é levantada se qualquer dos operandos for um NaN;
- boolean ge(float x, float y): retorna verdadeiro se x é maior que ou igual a y, e falso caso contrário;
- boolean gt(float x, float y): retorna verdadeiro se x é maior que y, e falso caso contrário;
- boolean le(float x, float y): retorna verdadeiro se x é menor que ou igual a y, e falso caso contrário;
- boolean leQuiet(float x, float y): retorna verdadeiro se x é menor que y, e falso caso contrário. NaNs não causam uma exceção;
- boolean lt(float x, float y): retorna verdadeiro se x é menor que y, e falso caso contrário;
- ltQuiet(float x, float y): retorna verdadeiro se x é menor que y. NaNs não causam uma exceção.

Controle de Arredondamento

O padrão IEEE-754 estabelece que toda operação será executada como se fosse primeiro produzido um resultado intermediário correto para exatidão infinita e com faixa ilimitada, e então arredonda o resultado de acordo com um dos quatro modos de arredondamento.

A maioria de aplicações numéricas requer operações de ponto flutuantes que usam só o modo padrão de arredondamento do IEEE-754 (arredondamento para o mais próximo). Porém, há aplicações que precisam de outros modos de arredondamento.

JFloat provê todos os quatro modos de arredondamento como definido no padrão de IEEE-754.

A biblioteca JFloat define os quatro modos de arredondamento definidos pela norma, e os deixa disponíveis a partir das constantes públicas:

- public static final int ROUND_NEAREST_EVEN = 0;
- public static final int ROUND_DOWN = 1;
- public static final int ROUND_UP = 2;
- public static final int ROUND_TO_ZERO = 3;

O modo de arredondamento atual afeta o resultado da operação como definido no padrão de IEEE-754.

Pode-se especificar e inspecionar o modo de arredondamento atual, respectivamente, usando as funções:

- void setRoundingMode(int roundingMode): essa função define o arredondamento desejado para as operações matemáticas subseqüentes.
- int getRoundingMode(): retorna o modo de arredondamento atual.

No código abaixo, temos o exemplo de uma adição intervalar usando JFloat:

Tabela 2: Adição intervalar.

```
float a_low, a_high,
      b_low, b_high,
      c_low, c_high;

a_low = 0.3f ; a_high = 0.5f;
/* a = [0.3, 0.5] */

b_low = 0.25f; b_high = 0.75;
/* b = [0.25, 0.75] */

/* Arredondamento para baixo do limite inferior
*/
JFloat32.setRoundingMode(ROUND_DOWN);
c_low = JFloat32.add(a_low, b_low);

/* Arredondamento para cima do limite superior
*/
JFloat32.setRoundingMode(ROUND_UP);
c_high = JFloat32.add(a_high, b_high);

/* c = [0.3 + 0.25, 0.5+0.75] */
```

Testes

Foram realizados testes com a biblioteca JFloat32 comparando-a com o tipo nativo de precisão simples float e o tipo nativo de precisão dupla double. Como padrão de referência foi usado o tipo de dupla precisão da biblioteca C-XSC[2].

Os testes foram realizados em um computador Toshiba, modelo Satellite, com processador Intel Centrino Core Duo, de 1,6 GHz com 1Gb de RAM e 60 Gb de disco rígido. A máquina virtual usada nos testes foi a JVM 5.0 da Sun.

No primeiro teste, vide tabela abaixo, foi verificada a velocidade da função seno de JFloat comparada com a função nativa de dupla precisão da linguagem Java e a função seno da biblioteca C-XSC.

Tabela 3: Comparação da velocidade da função seno.

Biblioteca	Tempo (ms)
Java	1.014
C-XSC	5.397
JFloat	16.036

No segundo teste, foi verifica a exatidão de cada operação de JFloat32, float e double nativos de Java em relação ao tipo de ponto de flutuante de dupla precisão da biblioteca C-XSC.

Tabela 4: Exatidão do tipo nativo float e JFloat em relação a C-XSC.

	Tipo	Erro Maximo	Erro Médio	Desvio padrão
+	JFloat	1,1757E-07	4,2291E-08	8,9198E-09
	Float	5,9602E-08	4,2276E-08	8,8335E-09
-	JFloat	1,7849E-07	4,2246E-08	8,9919E-09
	Float	5,9601E-08	4,2232E-08	8,8448E-09
*	JFloat	6,9948E-06	4,2422E-08	1,5898E-07
	Float	6,9947E-06	4,1993E-08	1,5898E-07
/	JFloat	7,8868E-06	4,4152E-08	3,3879E-08
	Float	7,8868E-06	4,3727E-08	3,3861E-08

No último teste, para as operações de adição e subtração, verificou-se que o arredondamento da biblioteca JFloat obteve resultados mais próximos do valor obtido pela biblioteca C-XSC.

Verificamos que apesar da implementação nativa de Java ser mais precisa que a presente em JFloat, o percentual apresentado foi irrisório e esperado, uma vez que internamente Java trabalha com precisão dupla, inclusive para float e ao final faz o arredondamento.

Conclusões

Como esperado, as operações aritméticas executadas por JFloat serão mais lentas que a executada por um hardware dedicado com um processador numérico, ou mesmo as operações aritméticas definidas em Java.

Todavia, como atualmente inexiste controle nativo direcionado de arredondamento em Java, JFloat é uma opção para aplicações que precisam desse controle e que possam ignorar o hardware e fazê-lo completamente por software.

Testes feitos com a biblioteca, quando comparadas aos tipos de ponto flutuante de precisão simples nativo de Java e o definido na biblioteca C-XSC, mostraram que suas exatidões são muito próximas.

Referências

[1] Boisvert, R. F. e J. P. M. P. R. Moreira. Java and numerical computing. Computing in Science Engineering, v.3, n.2, p.18-24. 2001.

[2] Hofschuster, K. C-XSC 2.0: a C++ library for extended scientific computing. Heidelberg: Springer-Verlag, v.2991. 2004

[3] IEEE. ANSI/IEEE Standard for Binary Floating-Point Arithmetic. ANSI/IEEE. New York. 1985.

[4] Kahan, W. How java floating-point hurts everyone everywhere. http://cch.loria.fr/documentation/ IEEE754/ wkahan/JAVAHurt.pdf