Otimização de Rotinas e Funções da Biblioteca C-XSC[1]

Felipe Nardon[2], Eli Maruani[2], Carlos Amaral Hölbig

Curso de Ciência da Computação – ICEG
Universidade de Passo Fundo (UPF)
99.001-970 – Passo Fundo – RS – Brasil
{97548,91049,holbig}@upf.br

Resumo. *Este trabalho apresenta algoritmos e técnicas alternativas que objetivam um acréscimo de desempenho para programas que visam a alta exatidão escritos com a biblioteca C-XSC. Para viabilizar de forma eficiente essas alternativas foi realizado um estudo e apresentado um algoritmo otimizado para o cálculo do produto escalar entre vetores de números em ponto-flutuante com múltipla precisão, uma técnica de otimização que realiza a troca do modo de arredondamento em operações sobre intervalos e uma forma de otimizar programas que utilizam a biblioteca C-XSC para a resolução de cálculos comuns da álgebra linear que são muito utilizados em aplicações reais de grande porte.*

1. Introdução

O desempenho computacional obtido pela biblioteca C-XSC nas mais diversas implementações demonstra a necessidade do estudo de novas alternativas de otimização para suas rotinas básicas. Um C-XSC mais eficiente em termos de desempenho computacional possibilitaria sua efetiva utilização na resolução de aplicações reais de grande porte que necessitassem de uma melhor exatidão (difícil de alcançar com as ferramentas computacionais tradicionais). Exempos de aplicações reais que fazem uso do C-XSC podem ser encontradas em http://www.cs.utep.edu/interval-comp/main.html. Entre estas alternativas pode-se citar a implementação de novos algoritmos (rápidos) de somatório e/ou de produto escalar, de alterações na forma de implementação das suas atuais rotinas, da inclusão em suas rotinas de funções da BLAS (*Basic Linear Algebra Subroutines*), do uso das funções SSE (*Streaming* SIMD *Extensions*) e de diretivas de otimização do compilador *gcc*. O objetivo deste trabalho é identificar possíveis gargalos de desempenho na biblioteca C-XSC e propor técnicas e algoritmos numéricos rápidos que possam ser utilizadas como alternativas para aumentar o desempenho dos programas com alta exatidão implementados nesta biblioteca. Para isto, este trabalho propõe alternativas otimizadas para três tipos de operações da biblioteca C-XSC: a troca do modo de arredondamento nas operações intervalares, o produto escalar calculado através da função `accumulate` e alguns cálculos matemáticos específicos da álgebra linear.

[1] Projeto de pesquisa financiado pelo CNPq (processo 475425/2009-0)

[2] Bolsista PIBIC-CNPq, [2] Bolsista BIC-FAPERGS

2. Cálculo do Produto Escalar Através do Algoritmo DotK

O algoritmo `DotK` [Ogita 2005] é uma alternativa para o cálculo do produto escalar com múltiplo grau de precisão. Este algoritmo faz uso de uma técnica comumente chamada de *Error Free Transformation* (EFT). Através desta técnica, é possível adquirir a estimativa do erro gerado em determinadas operações aritméticas. Desta forma, um grau maior de exatidão é conferido ao resultado simplesmente somando-o com a estimativa do erro calculada. Segundo Ogita (2005), a utilização de variáveis com precisão extra (como é o caso do C-XSC) é um fator que, além de reduzir o desempenho de aplicações de forma significante, restringe a aplicação a computadores e compiladores específicos. Em contraste a isto, o algoritmo `DotK` é capaz de realizar o cálculo do produto escalar com alta exatidão utilizando-se apenas de operações em aritmética convencional (ordinária), visto que os algoritmos para soma e multiplicação por ele utilizados também estão implementados apenas com este tipo de operação. Resultados obtidos por meio da implementação destes algoritmos podem ser encontrados em [Zimmer 2008].

3. Otimizações da Troca do Modo de Arredondamento em Operações sobre Intervalos

Segundo Bohlender (2002), a aritmética intervalar, implementada no C-XSC para garantir a aplicação da Validação Numérica, é mais lenta e demanda mais computações do que a aritmética convencional. Isto vem do fato de que um intervalo é constituído de no mínimo dois valores, um extremo inferior e um extremo superior, e as operações devem ser feitas levando-se em conta os dois extremos. Além disso, toda e qualquer operação sobre intervalos necessita de computações adicionais para o ajuste do arredondamento dos resultados. Uma típica operação de adição entre estes intervalos é realizada conforme (1) armazenando-se o resultado em outro intervalo ***S***.

$$S = A + B = [\nabla(a.inf + b.inf), \Delta(a.sup + b.sup)] \quad (1)$$

Na equação, ∇ representa que o resultado da operação (***a.inf + b.inf***) será arredondado para baixo, ou seja, para o maior número representável pela máquina que seja menor ou igual ao próprio resultado. Da mesma forma, Δ representa que o resultado da operação (***a.sup + b.sup***) será arredondado para cima, ou seja, para o menor número representável pela máquina que seja maior ou igual ao próprio resultado.

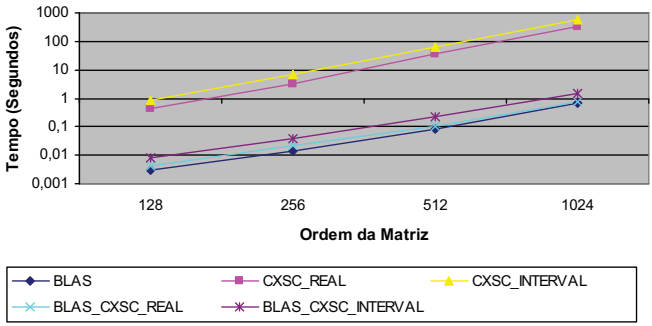
Uma forma de otimizar a troca do modo de arredondamento em operações sobre intervalos foi proposta por Bohlender (2002). A idéia básica é a de usar negações ao invés de trocas explícitas no modo de arredondamento. Desta forma, é possível estabelecer a seguinte igualdade: $\Delta(a) = -\nabla(-a)$. Por exemplo, suponha que $a = 0{,}3$ e que a é um número computacionalmente representável, que o arredondamento da máquina está ajustado para baixo e que, em função disto, aplicando-se $\nabla(a)$ tem-se o valor $0{,}3$. Desta forma, é possível encontrar o valor de $\nabla(a)$ através de duas operações sem trocar o modo de arredondamento para cima. Utilizando esta técnica, a soma descrita em (1) poderia ser realizado como descrito em (2), trocando o modo de arredondamento para baixo antes da realização de primeira operação.

$$S = A + B = [(a.inf + b.inf), -((-a.sup) + (-b.sup))] \quad (2)$$

Esta redução no desempenho pode ser ainda maior quando as operações intervalares são realizadas em vetores ou matrizes de intervalos. Neste caso, sendo ***n*** o número de elementos de um vetor, seria necessário um mínimo de ***2n*** trocas no modo de arredondamento para uma simples soma entre vetores, por exemplo. Utilizando a técnica de negação, com apenas uma troca no modo de arredondamento é possível realizar todos os cálculos.

4. Integração do C-XSC com Rotinas de Bibliotecas de Alto Desempenho

Um tópico importante no que tange a otimização de funções e rotinas da biblioteca C-XSC é sua integração com rotinas/funções de bibliotecas de alto desempenho como, por exemplo, BLAS, LAPACK e ScaLAPACK, tanto para programas sequenciais como paralelos. Esta abordagem está propiciando uma melhora considerável do desempenho do C-XSC, em especial em rotinas que abordam o cálculo do produto escalar e da multiplicação de matrizes. Neste caso, o uso das rotinas `dot` e `gemm` da BLAS demonstraram um ganho considerável de desempenho, mesmo quando adaptadas para uso com dados intervalares, conforme apresentado na Figura 1. No caso dos *solvers* sequenciais e paralelos para a resolução de sistemas de equações lineares, o uso de rotinas de triangularização e de decomposição LU da LAPACK e ScaLAPACK estão propiciando uma melhora no desempenho destes *solvers*. Resultados mais detalhados sobre o uso destas abordagens podem ser encontrados em [Zimmer and Krämer, 2008], [Kolberg, Fernandes and Claudio, 2008], [Almeida and Hölbig, 2009] e [Lima and Hölbig, 2009].



5 Conclusões e Trabalhos Futuros

A pesquisa na qual este trabalho está inserido procura demonstrar que a questão de erros gerados em cálculos sobre números em ponto-flutuante, realizados em um computador, deve ser levada em conta quando se trabalha com Computação Científica e, também, que deve-se dar especial importância ao nível de otimização empregado nos algoritmos que solucionam este tipo erro, a fim de evitar que se crie um novo problema, relacionado ao desempenho computacional. Levando-se em conta esta afirmação, este

artigo apresentou algoritmos e técnicas que podem ser utilizadas para garantir um ganho de desempenho em algumas operações da biblioteca C-XSC.

Neste sentido este texto apresentou uma técnica otimizada para a troca no modo de arredondamento em operações sobre intervalos, baseada nos estudos de Bohlender (2002), onde, com base nos resultados apresentados, pode-se concluir que esta técnica é viável, tanto na questão da exatidão do resultado quanto na questão do aumento de desempenho que a mesma proporciona. No decorrer da pesquisa foi possível constatar, também, que quanto maior o grau de exatidão empregado na solução de um problema, maior é o custo computacional para a sua resolução. Desta forma, a escolha deste grau deve ser feita levando-se em conta o problema em específico que se deseja solucionar, para que se possa ter um resultado útil em um tempo aceitável. Com a utilização do algoritmo `DotK` foi possível determinar em tempo de execução o nível de exatidão desejado e, como consequência, o nível de desempenho desejado. Desta forma, podem-se levar em conta estas duas variáveis para o problema em específico que se está resolvendo. Esta alternativa, ao contrários das demais apresentadas neste artigo, gera uma perda de exatidão em prol do ganho de desempenho, conforme demostrado por Zimmer e Krämer (2008). Outra forma de otimizar programas desenvolvidos com a biblioteca C-XSC aborda a integração de bibliotecas externas ao C-XSC para realizar operações aritméticas básicas da álgebra linear, tais como a BLAS e a LAPACK. Outro fator que pode contribuir com a otimização da biblioteca C-XSC é o uso de instruções em linguagem de montagem. Através destas é possível otimizar rotinas específicas além da utilização de instruções alternativas e otimizadas para cálculos em ponto-flutuante, apresentadas nos conjuntos SSE, MMX e 3DNow.

Referências Bibliográficas

Almeida, A. and Hölbig, C. (2009) “Paralelização do *Solver* Intervalar LSS”, In *9ª Escola Regional de Alto Desempenho*, SBC/UCS/UFSM/UFRGS/UPF, Porto Alegre, p. 221-224.

Bohlender, G. (2002) Faster interval computations through minimized switching of rounding modes, http://www.math.uni-wuppertal.de/~xsc/preprints/, Abril, 2008.

Grimmer, M. (2008) “An MPI extension for the use of C-XSC in parallel environments”, http://www.math.uni-wuppertal.de/~xsc/preprints/, Abril.

Kolberg, M. L., Fernandes, L. G. and Claudio, D. M. (2008). Dense Linear System: A Parallel Self-verified Solver. In: *International Journal of Parallel Programming*, v. 36, pages 412–425.

Lima, A. A. M. and Hölbig, C. (2009) “Paralelização do *Solver* Intervalar para a Resolução de Sistemas Lineares Esparsos”, In *9ª Escola Regional de Alto Desempenho*, SBC/UCS/UFSM/UFRGS/UPF, Porto Alegre, p. 213-216.

Ogita, T., Rump, S. M. and Oishi, S. (2005). Accurate sum and dot product. In *SIAM Journal of Scientific Computing*, v. 26, pages 1955–1988.

Zimmer, M. and Krämer, W. (2008). Fast (parallel) dense linear interval systems solvers in C-XSC using error free transformations and BLAS. In *Numerical Validation in Current Hardware Architectures*. Dagstuhl: Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), (Dagstuhl Seminar Proceedings, 08021).