Análise de Desempenho do Solver Verificado LSS

Alexandre Almeida[1], Carlos Amaral Hölbig[1]

[1]Curso de Ciência da Computação – Universidade de Passo Fundo (UPF)
Campus 1 – BR 285 – Bairro São José – CEP 91501-970 – Passo Fundo – RS – Brasil
{68935,holbig}@upf.br

Resumo. *Este artigo apresenta uma análise do desempenho dos produtos escalares e da inversão de matrizes do solver verificado LSS, com o objetivo de, em um trabalho futuro, propor e implementar uma nova estratégia de paralelização para este solver. O LSS é utilizado para a resolução de sistemas densos de equações lineares do tipo Ax = b, através do uso da biblioteca C-XSC, que garante o tratamento da instabilidade numérica na aritmética de ponto-flutuante, por meio do emprego do paradigma da Computação Verificada.*

1. Introdução

De acordo com [Cláudio; Marins *apud* Do Carmo 2006], a resolução de sistemas de equações lineares do tipo $Ax = b$ é um dos assuntos mais abordados em Análise Numérica, sendo ainda pertinente à Computação Científica, no que diz respeito tanto à qualidade numérica dos resultados, quanto ao tempo de resposta dos programas.

A qualidade numérica pode ser obtida através do uso da biblioteca C-XSC, que possibilita a utilização do paradigma da Computação Verificada, substituindo a aritmética ordinária realizada pelo processador, por uma aritmética intervalar e de alta exatidão, realizada via software, para a resolução das operações matemáticas.

Através da biblioteca C-XSC, o *solver* LSS (*Linear System Solver*) tem como objetivo a resolução de sistemas lineares densos utilizando o paradigma da Computação Verificada. Embora a qualidade numérica dos resultados possa ser garantida pelo uso da Computação Verificada, há uma notável perda em desempenho neste *solver*, devido à necessidade do C-XSC processar os tipos de dados de alta exatidão via software, como, por exemplo, na operação de variáveis intervalares, onde deve-se considerar ambos limites inferior e superior em cada cálculo [Grimmer 2005].

Em vista do baixo desempenho da versão seqüencial do *solver* LSS, Do Carmo (2006) realiza um estudo detalhado do funcionamento deste, abrangendo suas características, rotinas e algoritmos, propondo e implementando ainda uma versão paralela [Do Carmo 2006, p. 71]. Devido ao baixo ganho em desempenho apresentado pela versão paralela desenvolvida pela autora [Do Carmo 2006, p. 84], este trabalho tem como objetivo identificar pontos críticos na versão seqüencial do *solver*, a fim de, futuramente, propor e implementar uma nova estratégia de paralelização.

2. Solver LSS

O *solver* verificado LSS, desenvolvido com o uso da biblioteca de alta exatidão C-XSC, tem por objetivo a resolução de sistemas de equações lineares do tipo $Ax = b$, sendo A

uma matriz densa de coeficientes; *x* uma matriz de incógnitas; e *b* uma matriz de termos independentes.

Portanto, o *solver* é capaz de tratar sistemas de equações lineares com matrizes do tipo densa, auxiliando na resolução de sistemas lineares sobre-determinados ($m \times n$, com $m > n$); sistemas lineares quadrados ($n \times n$); sistemas lineares sub-determinados ($m \times n$, com $m < n$); cálculo da inversa A^{-1} de A (no caso de sistemas quadrados); cálculo da pseudo-inversa A^{+} de A (no caso de sistemas sobre-determinados); e cálculo da pseudo-inversa A^{+} de A (no caso do sistemas sub-determinados) [Hölbig 2005, p. 46].

3. Solver LSS Paralelo

A versão paralela do *solver* LSS desenvolvida por Do Carmo (2006) explora o paralelismo do cálculo de todos os produtos escalares realizados no programa, implementando funções para tal propósito, as quais suportam os tipos de dados de alta exatidão do C-XSC, ou seja, a qualidade do resultado foi mantida também na versão paralela.

Entretanto, como pode ser observado pelo gráfico da Figura 1, quando executado em paralelo, o *solver* sofre um *slowdown* com relação à sua versão seqüencial.

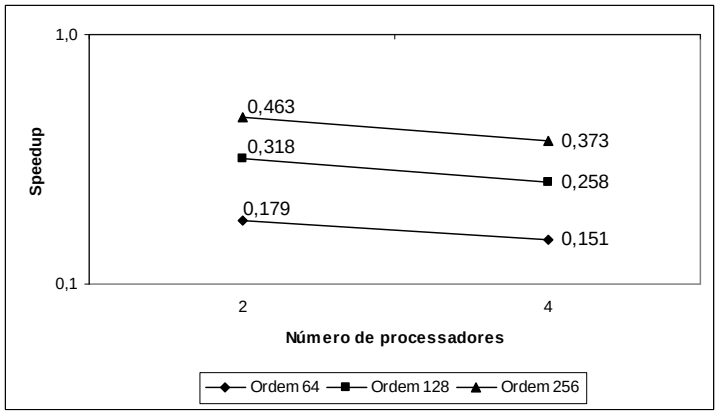

Figura 1. Speedup do *solver* LSS paralelo

Tal perda em desempenho pode decorrer do fato de que todas as etapas do programa que realizavam produtos escalares foram paralelizadas, sendo que, para determinados casos, a paralelização de um produto escalar não é necessária.

4. Testes de desempenho

4.1. Metodologia

Os testes de desempenho no *solver* seqüencial foram realizados utilizando-se sistemas de entrada na ordem de 64, 128, 256, 512 e 2048 elementos. Os trechos de código considerados nos testes foram todos aqueles que realizavam o cálculo do produto escalar e o cálculo da inversa dos sistemas. Vale ressaltar ainda que o cálculo da inversa é efetuado através do método de Gauss-Jordan. Para os sistemas de 64 e 128 elementos, foi feita uma média aritmética de 100 repetições, sendo que para os sistemas de 256 e

512 foram consideradas 10 repetições. Para o sistema de 2048 elementos, a média não foi calculada devido ao alto tempo de processamento obtido para tal magnitude.

Todas as execuções foram realizadas em um ambiente computacional com um processador Intel Celeron M, à freqüência de 1.6 GHz, e com 512 MB de memória principal. O sistema operacional utilizado foi o GNU/Linux, kernel 2.6 (distribuição Debian 4).

4.2. Resultados

Com relação aos resultados obtidos, foi constatado que alguns dos produtos escalares possuem um desempenho razoável na sua forma seqüencial, fazendo com que o custo da sobrecarga da paralelização tenda a causar uma perda no desempenho, não só do produto escalar em sua forma isolada, mas no *solver* como um todo.

Entretanto, determinados produtos escalares sofrem um aumento considerável no tempo de processamento a medida que os sistemas de entrada aumentam. Para tais trechos de código, a paralelização pode ser uma alternativa.

A Figura 2 ilustra o comportamento do aumento do tempo de processamento dos produtos escalares do *solver*, em função do tamanho de um sistema de entrada na forma $Ax = b$. As séries de dados de 1 a 5 ilustram o tempo de processamento para o cálculo de uma matriz inversa de tamanho simples, processamento este que envolve a determinação de uma aproximação de x (série 1), o cálculo do erro arredondado em precisão dupla e em alta exatidão (séries 2 e 3, respectivamente), e os cálculos da inclusão do resíduo de x, e a determinação de uma matriz de iteração (séries 4 e 5, respectivamente).

As séries de 6 a 13 representam o cálculo da inversa de x de tamanho duplo, caso a solução exata não tenha sido obtida para uma inversa de tamanho simples. Tal processamento envolve o cálculo do erro entre a multiplicação das inversas de tamanho simples e de tamanho duplo (série 6), a determinação de um resultado parcial b e o cálculo do erro em alta exatidão do resultado parcial (séries 7 e 8, respectivamente). As séries de 9 a 13 representam os mesmos procedimentos das séries de 1 a 5 [Do Carmo 2006, p. 37], porém sempre considerando a inversa de tamanho duplo.

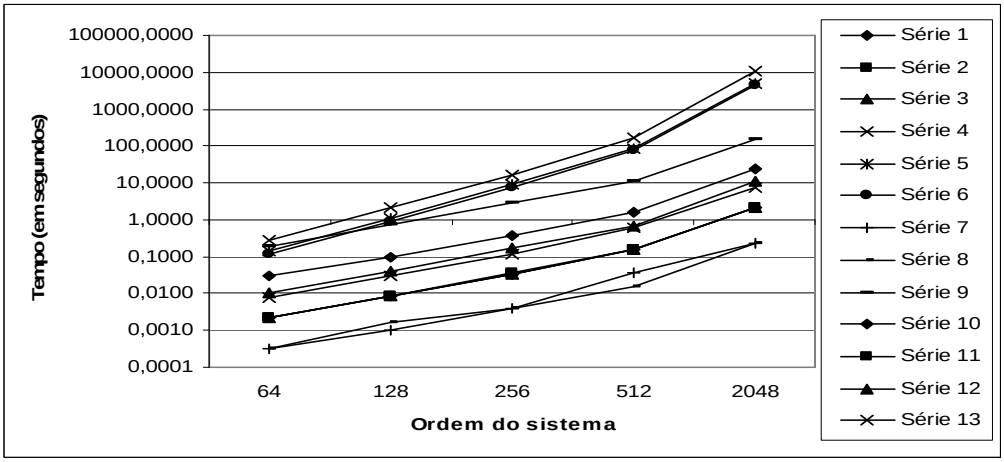

Figura 2. Crescimento dos tempos de processamento dos produtos escalares

Em relação ao comportamento do aumento do tempo de processamento dos métodos de inversão de matrizes (tanto para precisão simples, quanto para dupla precisão), representado pela Figura 3, pode-se notar um crescimento quase exponencial a medida que a ordem do sistema aumenta, fato que torna o cálculo da inversa um candidato a ganhar uma versão paralela.

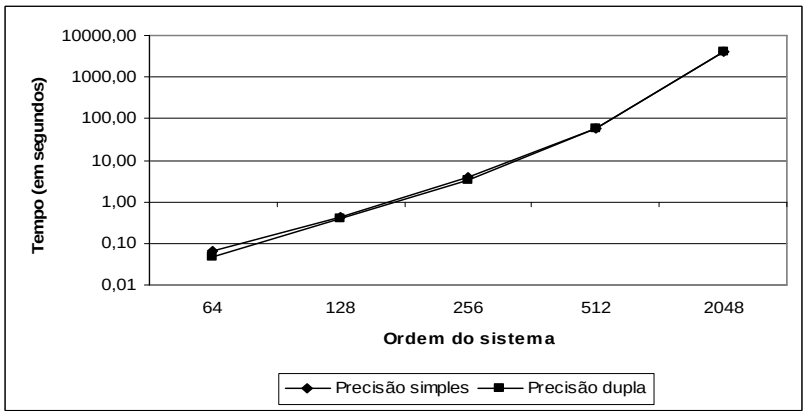

Figura 3. Crescimento dos tempos de processamento para os cálculos da inversa

5. Considerações finais

De acordo com os resultados obtidos, pode-se concluir que nem todos os produtos escalares são bons candidatos a serem paralelizados. Caso o algoritmo do produto escalar possua um desempenho razoável em sua forma seqüencial (por exemplo, àqueles que tendam a situarem-se próximos ou abaixo de 1 segundo pela Figura 2), a paralelização deste tende apenas a aumentar o tempo de processamento do *solver*, devido ao alto custo da comunicação entre os nodos de processamento, e da sobrecarga introduzida pelas funções para o processamento paralelo.

Além disso, foi possível identificar que o método responsável pelo cálculo da inversão de matrizes é uma das etapas do *solver* que mais necessita de tempo de processamento, o que configura este como sendo um bom candidato a ser executado em paralelo.

Referências

Do Carmo, Andriele Busato. (2006) "Paralelização de Solver Intervalar para a Resolução de Sistemas Lineares Densos". Trabalho de Conclusão II (Graduação em Ciência da Computação) – Universidade de Passo Fundo, Passo Fundo.

Grimmer, Markus. (2005) "An MPI Extension for the Use of C-XSC in Parallel Environments", In: WRSWT 2005/3. *Annals*… Wuppertal, Üniversität Wuppertal.

Hölbig, Carlos Amaral. (2005) "Ambiente de Alto Desempenho com Alta Exatidão para a Resolução de Problemas". Tese (Instituto de Informática/Programa de Pós-Graduação em Computação) – Universidade Federal do Rio Grande do Sul, Porto Alegre.