Utilização da BLAS em algoritmos otimizados para Cluster

Rúbia Medianeira Denardi[1]
Diego Fraga Contessa[2]
Tiarajú Asmuz Diverio[3]

Resumo

Este artigo apresenta os primeiros resultados de um estudo sobre o uso da biblioteca BLAS (Basic Linear Algebra Subprograms) no ambiente de máquinas de alto desempenho do tipo Cluster. Busca-se relatar as diferentes implementações e o uso eficiente destas rotinas na resolução de problemas numéricos, via programação paralela baseada em memória compartilhada e em troca de mensagens. Para demonstrar o uso e eficiência da BLAS, será apresentada a implementação do Método do Gradiente Conjugado e uma comparação de seu desempenho quando implementado com e sem o uso da BLAS.

1 Introdução

Com o avanço do uso de computadores de alto desempenho e o surgimento das máquinas vetoriais, das máquinas com memória virtual e das máquinas paralelas com memória compartilhada, tornou-se necessário utilizar técnicas que explorem esses aspectos da sua arquitetura, visando obtenção de desempenho na execução de suas tarefas. Algumas bibliotecas matemáticas são usadas para melhorar o desempenho de aplicações, principalmente na computação científica, como é o caso da BLAS (Basic Linear Algebra Subprograms).

2 BLAS (Basic Linear Algebra Subprograms)

A BLAS é uma biblioteca matemática com rotinas desenvolvidas em blocos, para resolver operações básicas da álgebra linear, que são fundamentais para cálculos na computação científica. Outra atribuição importante da BLAS é no projeto de computadores, na implementação de operações de baixo nível, como produto escalar e produto matriz-vetor. A biblioteca trabalha com matrizes dos tipos esparsas, de banda e densa.

[1]rmdenardi@inf.ufrgs.br Bolsista DELL
[2]contessa@inf.ufrgs.br Bolsista CNPq
[3]Apoio: LabTeC UFRGS/Dell e CNPq

A BLAS foi criada com o objetivo de incentivar a programação estruturada e melhorar a qualidade de documentação do software, pois especifica e identifica os blocos básicos de operações. A biblioteca está divida em três níveis de operações. O primeiro nível, BLAS1, é uma especificação e implementação de subprogramas para operações escalares e vetoriais. Com o aparecimento das máquinas vetoriais, especificações para a BLAS2 e BLAS3 foram implementadas para operações matriz-vetor e matriz-matriz respectivamente, tornando possível uma utilização mais eficiente dos computadores modernos (DONGARRA, 2001).

Na BLAS1 existem basicamente operações de redução de escalares e vetores, rotação de vetores, operações vetoriais e vetoriais com movimentos de dados.

Tabela 1. Algumas Operações da BLAS1

Dot product	$r \leftarrow \beta\, r + \alpha x^T y$
Sum	$r \leftarrow \sum_i x_i$
Scaled vector accumulation	$y \leftarrow \alpha x + \beta y,$
Apply plane rotation	$(x \quad y) \leftarrow (x \quad y)R$

A BLAS2 trabalha com operações entre vetores e matrizes. Em computadores modernos, a relação entre operações de ponto-flutuante e movimento de dados é muito baixa para conseguir o uso efetivo da memória cache. O nível 2 foi projetado para otimizar o reuso dos dados em registradores e a redução do acesso a memória.

Tabela 2. Algumas Operações da BLAS2

Matrix-vector product	$y \leftarrow \alpha\, Ax + \beta y, \quad y \leftarrow \alpha\, A^T x + \beta y$
Summed matrix-vector multiplies	$y \leftarrow \alpha Ax + \beta Bx$
Triangular solve	$x \leftarrow \alpha\, T^{-1} x, \quad x \leftarrow \alpha\, T^{-T} x$

Para a BLAS3 foram desenvolvidas as operações entre matrizes (operações algébricas de maior custo computacional), com o intuito de tirar vantagem das arquiteturas modernas. A BLAS3 possibilta a maximização da relação de operações de ponto-flutuante às referências a memória, onde dados residentes na memória cache ou local são reusados por quebras de matrizes em pequenos blocos.

Tabela 3. Algumas Operações da BLAS3

Matrix add and scale	$B \leftarrow \alpha A + \beta B, \quad B \leftarrow \alpha\, A^T + \beta B$
Matrix-matrix product	$C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha A^T B + \beta C$ $C \leftarrow \alpha AB^T + \beta C, C \leftarrow \alpha A^T B^T + \beta C$ $C \leftarrow \alpha AB + \beta C, C \leftarrow \alpha A^T B + \beta C$
Triangular solve	$B \leftarrow \alpha\, T^{-1} B, \quad B \leftarrow \alpha\, T^{-T} B$ $B \leftarrow \alpha\, BT^{-1}, \quad B \leftarrow \alpha\, BT^{-T}$

Existe também um conjunto de rotinas paralelas para resolver problemas que envolvam álgebra linear chamada de BLAS Paralela. Os pacotes de BLAS paralela disponíveis para a comunicação entre processos são todos baseados nas rotinas da BLAS, e entre eles estão:

- BLACS (Basic Linear Algebra Communication Subprograms) - biblioteca de troca de mensagens que inclui rotinas síncronas send/receive para comunicar uma matriz ou submatriz de um processo para outro, para broadcast de submatrizes para muitos processos, ou para computar reduções de dados globais (CORPORATION, 2001);

- PBLAS (Parallel Basic Linear Algebra Subprograms) - biblioteca com funcionalidade similar a da BLAS. Faz uma paralelização simplificada dos códicos densos da álgebra linear, construindo blocos paralelos (CORPORATION, 2001);

- ScaLAPACK (Scalable Linear Algebra Package) - esta biblioteca inclui um subconjunto das rotinas da LAPACK, reprojetado para computadores paralelos com memória distribuída, MIMD. É projetada para computação heterogênea e é portátil em todo o computador que suportar MPI ou PVM. As rotinas do ScaLAPACK são baseadas em algoritmos de blocos particionados, sendo os fundamentais: versão para memória distribuída (PBLAS) dos níveis 1, 2 e 3 da BLAS e um conjunto de rotinas de comunicação de álgebra linear (BLACS) para tarefas de comunicação que ocorrem freqüentemente em computações paralelas da álgebra linear. Nas rotinas da ScaLAPACK, toda a comunicação entre processadores ocorre dentro da PBLAS e da BLACS (BLACKFORD; CLEARY; DEMMEL, 2001).

3 Programação em Clusters de Computadores

As aplicações nas quais as rotinas da BLAS são tipicamente usadas necessitam, essencialmente, de alto desempenho de execução. Assim, é necessário notar que a BLAS foi desenvolvida inicialmente para uso em máquinas vetoriais e supercomputadores, os quais têm alto custo e, por esse motivo, não são acessíveis para a maioria dos usuários. Com a diminuição progressiva do custo dos computadores pessoais, fato que se acentuou nos últimos anos, popularizou-se a construção e utilização de Clusters (Agregados) de computadores. Essas máquinas têm sido vistas como uma solução para universidades e empresas que não possuem os recursos para a compra de supercomputadores (CARVALHO, 2002).

Um Cluster de Computadores, segundo Buyya (BUYYA, 1999), consiste de uma coleção de computadores interconectados, operando em conjunto para a execução de programas. Nesse tipo de máquina, cada nodo é um computador completo, com um ou mais processadores e memória privada. Assim, a comunicação entre os nodos é feita via rede de interconexão. Além disso, um cluster pode ser homogêneo, desde que seus nodos possuam as mesmas características de hardware e software, ou heterogêneo, caso contrário. A visão geral da arquitetura de um cluster de alto desempenho pode ser vista na Figura 1.

Para que se tire proveito das características de processamento paralelo que um cluster de computadores pode agregar às aplicações, é necessário conhecer as técnicas de programação aplicáveis a esse tipo de máquina, bem como os paradigmas que podem ser utilizados. A

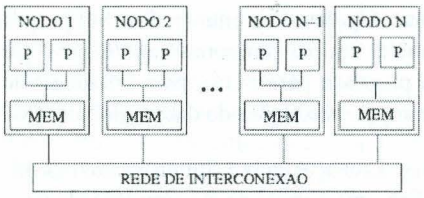

Figura 1: Arquitetura Geral de Cluters de Computadores

programação paralela em clusters de alto desempenho pode ser feita seguindo-se dois paradigmas, que diferem entre si pela forma como realizam a comunicação entre dois processos: a programação usando comunicação por memória compartilhada e a programação com comunicação por troca de mensagens.

A programação com comunicação por memória compartilhada consiste no aproveitamento da característica de que cada nodo é um sistema computacional completo, contendo memória privativa. Assim, diferentes processos no mesmo nodo podem se comunicar através da memória compartilhada entre eles, já que o espaço de endereçamento que eles acessam é o mesmo. De fato, processos em uma mesma máquina se comunicam usando as chamadas variáveis compartilhadas. Esse paradigma de programação paralela se mostra ainda mais interessante quando se trabalha com máquinas que possuem mais que um processador por nodo, o que é o caso de grande parte dos clusters de computadores atualmente existentes. Nesse caso, a comunicação entre processos distintos, executando em processadores diferentes do mesmo nodo, podem se comunicar como se estivessem sendo executados no mesmo processador, ou seja, usando a memória que os processadores de mesmo nodo compartilham. O ponto do sistema onde pode ocorrer um gargalo de desempenho é o barramento, que conecta os processadores com a memória, tendo em vista que o acesso ao barramento é seqüêncial.

A programação com comunicação por troca de mensagens se difere da anterior pelo fato de considerar a possibilidade de executar processos em processadores residentes em máquinas distintas dentro do agregado. Nesse quadro, eles não possuem memória compartilhada, o que significa que um não pode acessar o espaço de endereçamento que o outro acessa. Assim, para cada comunicação entre processos em máquinas distintas, é necessário o envio e recebimento de uma mensagem através da rede de interconexão. Um problema adicional deve ser evitado: a rede de interconexão deve ser rápida o suficiente para não se tornar um gargalo do sistema. Por outro lado, a aplicação deve utilizar coerentemente a rede de interconexão, de modo a evitar a sobrecarga da mesma, evitando que ela se torne um gargalo de desempenho. Nota-se que a escalabilidade de aplicações que se baseiam em comunicação por troca de mensagens é uma questão importante. Além disso, há ainda um overhead em ambos os lados da comunicação, decorrente das operações realizadas pelo protocolo de rede utilizado.

4 Aplicação e Testes

Primeiramente, foi instalada a biblioteca ATLAS, pacote free disponível no Netlib (NETLIB, 2003), com as rotinas BLAS serial. Estas rotinas foram testadas no Método do Gradiente Conjugado (CG) para uma comparação de desempenho entre o método original e o método alterado com a utilização das rotinas da BLAS.

O GC é um método de Resolução de Sistemas de Equações Lineares, iterativo não estacionário, pertencente à classe de métodos do subespaço de Krylov, e parte do princípio de que o gradiente, que é um campo vetorial, aponta sempre na direção mais crescente da função quadrática. O algoritmo do CG busca minimizar uma determinada função quadrática, de modo que se a matriz for SDP (Simétrica Definida Positiva) o gradiente dessa função $f(x)$ resume-se a $\bigtriangledown f(x) = b - Ax$. Então, como se deve minimizar o $\bigtriangledown f(x)$ deve-se determinar x tal que $\bigtriangledown f(x) = b - A.x = 0$, ou seja, $Ax = b$, significando que ao encontrar a solução do GC encontra-se a solução do sistema de equações.O algoritmo do GC é constituído basicamente de operações matriciais, como por exemplo, a soma e o produto escalar de vetores, e a multiplicação matriz por vetor. Essa última operação é a de maior custo computacional (SHEWCHUK, 1994).

Os testes foram feitos a partir da biblioteca ATLAS 3.2.1. As comparações aconteceram em nível de operações executadas com o uso da BLAS e sem o uso da biblioteca, como mostra a Tabela 4, onde são descritas algumas das operações contidas no GC e o tempo gasto (t) para cada operação. O algoritmo foi testado em um Pentium III 1.1 GHz, 1GBytes RAM, disco 20 GBytes, cache 512KBytes.

Tabela 4. Tempo de Execução do CG nos dois casos

Operações	t sem BLAS	t com BLAS
$d \Leftarrow r$	0.000001	0.000002
δ *novo* $\Leftarrow T^T T$	0.0000025	0.000003
$q \Leftarrow Ad$	0.000224	0.000032
$r \Leftarrow b - Ax$	0.000224	0.000032
$r \Leftarrow r$ - $\alpha\, q$	0.000003	0.000002
t total CG após 13 iterações	0.003243	0.000492

Foram usadas subrotinas da BLAS1 como *AXPY, DOT, COPY* e BLAS2 como *GEMV*. Os cálculos que exigiam um maior custo computacional, como a operação matriz-vetor, tiveram seu desempenho melhorado com o uso da BLAS.

5 Conclusão e Trabalhos Futuros

Do ponto de vista de programação, o uso das rotinas BLAS se mostra interessante, pois simplifica o trabalho do programador,visto que ele não precisa perder tempo programando

rotinas que já estão prontas, bastando usar as que são oferecidas pela BLAS. Mostrou-se importante o conhecimento a respeito da forma como a BLAS é implementada, para que se possa ter noção de como aproveitar as rotinas e otimizações da BLAS nos algoritmos que se desenvolve. Além disso, um estudo aprofundado a respeito das funções da biblioteca também é de grande valia ao trabalho.

No que diz respeito ao desempenho na execução dos programas, a BLAS mostrou-se mais eficiente, já que diminuiu o tempo de execução na resolução do método do Gradiente Conjugado. Os resultados positivos obtidos na comparação entre o desempenho do programa sem o uso da BLAS e o desempenho do programa que utiliza as rotinas da BLAS levam a crer que o uso coerente da BLAS em clusters de alto desempenho, levando em conta as diferenças entre as implementações da biblioteca e os paradigmas de programação paralela, trará um ganho de desempenho em algoritmos paralelizados. A mescla entre a utilização das rotinas da BLAS paralela e as rotinas seqüenciais da BLAS em algoritmos paralelizados cujo domínio é particionado também poderá trazer ganhos de desempenho. O uso da BLAS Paralela em Clusters será objeto de estudo na continuação da pesquisa.

Referências

BLACKFORD, S.; CLEARY, A. J.; DEMMEL, J. 2001. University of Tennessee. `http://www.netlib.org/scalapack/scalapack_home.html`. Acesso em: dezembro 2002.

BUYYA, R. (Ed.). *High Performance Cluster Computing: Architectures and Systems*. Upper Saddle River: Prentice Hall PTR, 1999. 849 p.

CARVALHO, E. C. A. *Particionamento de Grafos de Aplicações e Mapeamento em Grafos de Arquiteturas Heterogêneas*. Porto Alegre, 2002. Instituto de Informática. UFRGS. Dissertação de Mestrado.

CORPORATION, N. 2001. NEC Corporation. `http://www.support.ess.nec.de/mathkeisan120.html`. Acesso em: março 2003.

DONGARRA, J. et al. 2001. University of Tennessee. `http://www.netlib.org/utk/people/JackDongarra/papers.htm`. Acesso em: outubro 2002.

NETLIB. 2003. Netlib Repository at UTK and ORNL. `http://www.netlib.org/atlas`. Acesso em: agosto 2002.

SHEWCHUK, J. R. *An Introduction to the Conjugate Gradient Method without the Agonizing Pain*. São Paulo, 1994.