Análise quantitativa dos custos de comunicação para programas utilizando MPI executados em máquinas paralelas de memória distribuída

Corrêa, R.V.
INPE, SLB
ricardo@slb.inpe.br
Wuensche, C.A.
INPE, DAS
alex@das.inpe.br
Preto, A . J.
INPE, LAC
airam@lac.inpe.br
Stephany, S.
INPE, LAC
Stephan@lac.inpe.br

Resumo

O presente trabalho visa analisar os custos de comunicação numa maquina paralela de memória distribuida da Divisão de Astrofísica (DAS) do INPE, adquirida no escopo do projeto Temático FAPESP 2000/06770-2,e composta por 16 nodos baseados na arquitetura Intel IA-32 e utilizando o sistema operacional Linux. Os nodos estão interligados por um comutador e uma rede padrão Fast Ethernet. Foram utilizados programas teste envolvendo cálculos com vetores de pontos flutuantes, escritos em C e utilizando a biblioteca de comunicação por troca de mensagens MPI (Message Passing Interface). Esta biblioteca utiliza o protocolo TCP/IP para implementar a comunicação entre os nodos. As medidas mostram a parcela devida ao tempo de latência e ao tempo dependente da largura de banda em cada comunicação, os quais permitem estimar de maneira grosseira quais parâmetros influem na comunicação e como esta poderia ser otimizada.

1. Introdução

A necessidade de alto desempenho em aplicações científicas tem levado ao uso de arquiteturas paralelas. Destas, as que fornecem melhor custo-benefício são as máquinas paralelas de memória distribuída, que são também facilmente escaláveis e cada vez mais difundidas. Essas máquinas se compõem de nodos independentes ligados por uma rede de interconexão. Esses nodos são geralmente computadores produzidos comercialmente em larga escala, o que contribui para que os custos sejam baixos. O presente trabalho visa analisar os custos de comunicação numa máquina paralela de memória distribuída constituída por 16 nodos, cada nodo tem arquitetura Intel IA 32, processador Intel Pentium III de 800 Mhz, 128 Mbytes de memória principal e utiliza o sistema operacional Linux. Os nodos são interconectados por placas de rede padrão *Fast* Ethernet através de um comutador do tipo *store and forward*. Esta máquina paralela foi adquirida no escopo do projeto temático FAPESP 2000/06770-2 apresentado pela Divisão de Astrofísica (DAS) do INPE, tendo como objetivo paralelizar aplicações científicas na área de Astrofísica.

O desempenho de uma aplicação rodando numa máquina sequencial pode ser otimizado de forma a utilizar mais eficientemente o processador e a memória. Para tal, tanto o programador como o compilador contribuem para a geração de uma sequência de instruções em linguagem de máquina que maximizam a utilização dos *pipelines* do processador e utilizam a memória *cache* de maneira eficiente, minimizando acessos à memória principal e diminuindo o número de *page faults*, os quais implicam em acessos à área de *swap*.

Em máquinas paralelas de memória distribuída, segue-se frequentemente o paradigma SPMD (*Single Program, Multiple Data*), no qual cada nodo executa o mesmo programa em dados diferentes. As dependências entre dados alocados em processadores diferentes geram a necessidade de comunicação entre os mesmos. Este tempo de comunicação pode representar uma parcela considerável do tempo final de execução, levando a valores pobres de *speed-up,* que é a razão entre o tempo de execução sequencial e o tempo de execução paralela para um dado número de processadores. Deve-se, portanto, minimizar o tempo de comunicação para se obter bom desempenho na paralelização. A avaliação desses custos de comunicação discriminando as parcelas que os compõe é o escopo do presente trabalho.

Nos testes abordados, os programas foram paralelizados com o uso da biblioteca de comunicação por troca de mensagens MPI[1] *(Message Passing Interface*), a qual é implementada com o uso dos protocolos TCP/IP(*Transport Control Protocol/Internet Protocol*) e, nessa máquina paralela específica, do protocolo *Ethernet*.

O protocolo TCP é voltado para a conectividade e procura garantir o estabelecimento da conexão entre dois pontos, em que o lado receptor e o lado transmissor trocam mensagens até o encerramento da conexão. O processo da conexão entre dois nodos faz uso de rotinas do sistema

operacional as quais demandam tempo de processamento e exigem acessos à memória, competindo portanto pelo uso do processador e do sistema de memória com o programa paralelo em execução.

Os protocolos TCP/IP foram projetados para a interconexão remota de redes, e os pacotes de dados trafegam, tipicamente, através de várias redes, sendo roteados de uma para outra, por equipamentos específicos. A existência de congestionamentos nesses pontos de distribuição/roteamento pode levar à perda de pacotes e à sua consequente retransmissão. Esses protocolos incorporam algoritmos especiais para otimizar o tráfego de pacotes nos casos de congestionamento, mas que acabam onerando os custos de comunicação nas conexões como as estabelecidas na máquina paralela.

2. Protocolos TCP/IP e Fast Ethernet

A necessidade de troca de dados entre dois computadores exige que ambos se comuniquem utilizando a mesma linguagem, isto é, um mesmo protocolo. O estabelecimento de uma conexão entre dois pontos implica em troca de informações visando garantir a troca posterior de dados.

Os protocolos de rede fornecem justamente a sequência de procedimentos tomando a responsabilidade pela garantia da conexão e a formatação dos dados que serão trocados de forma a compatibilizá-los entre os tipos diferentes de máquinas que estão envolvidas na conexão.

No início do desenvolvimento das redes de computadores, as soluções de interconexão eram proprietárias, ou seja, determinada tecnologia só era suportada por um determinado fabricante. Assim, para facilitar a interconexão de sistemas de computadores, a ISO(*International Standards Organization*) desenvolveu um modelo de referência chamado OSI*(Open Systems Interconnection*), permitindo que os diversos fabricantes pudessem adaptar seus protocolos tendo esse modelo como ponto de partida para o desenvolvimento de protocolos de interconexão.

Os protocolos IEEE 802.2 e IEEE 802.3[2] conhecidos como padrão *Ethernet* são utilizados para estabelecer as conexões em grande parte das redes locais. Na máquina paralela em análise, utiliza-se a evolução desse padrão denominado padrão *Fast Ethernet.* Neste padrão, a taxa de transmissão de bits passou de 10 Mbits/s do padrão *Ethernet* para 100 Mbits/s. Os padrões *Ethernet* e *Fast Ethernet* possuem a mesma estrutura de montagem de seus pacotes, apenas a taxa de transmissão e a forma de modulação diferem entre os modelos.

O protocolo TCP permite que 1460 bytes de dados sejam transmitidos em cada pacote. Os cabeçalhos inseridos pelos protocolos TCP, IP e *Ethernet*, adicionam um total de 66 *bytes* montando um quadro máximo com 1526 *bytes*. Esses quadros ou pacotes circulam em rede passando por vários caminhos que podem estar congestionados, levando à não entrega ao destino especificado. Assim, foi inserido ao protocolo TCP, o controle de fluxo de quadros, adicionando restrições ao envio de pacotes visando diminuir o congestionamento, o que implica em aumento do custo da comunicação. Neste trabalho, observa-se a operação de dois desses algoritmos: Nagle e *Slow Start*.

O algoritmo de Nagle[3] foi adicionado ao TCP em 1984 através da RFC 896 (posteriormente RFC 1122). O algoritmo atua nos pacotes pequenos (inferiores à dimensão mínima disponível para transmissão), sempre que exista uma conexão TCP que não receba a confirmação do pacote através de um segmento ACK. O algoritmo coleta esses pacotes e os acumula para posterior envio.

Nos casos de transferência de dados, isso pode se tornar um problema, uma vez que pequenos pacotes podem ficar retidos até o completo preenchimento do campo de dados de um datagrama, implicando na diminuição da taxa de dados transmitidos e assim aumentando o custo da comunicação.

Após o estabelecimento de uma conexão, o lado que a solicita inicia a comunicação enviando pacotes múltiplos de dados na rede. A cada pacote enviado, o lado de destino dos pacotes deve responder com um pacote ACK. Um algoritmo denominado *Sliding Window*(Janela Deslizante)[4] permite que vários pacotes sejam enviados sem que se espere pelo retorno dos pacotes ACK de confirmação. Assim, o ajuste da quantidade de pacotes passíveis de envio permite que o lado que está enviando os dados, faça uma transmissão mais eficiente transmitindo mais pacotes numa rede não congestionada e menos pacotes numa rede congestionada.

O algoritmo *Slow Start*[5] foi implantado na camada TCP para melhorar o desempenho da comunicação realizando o controle de fluxo de pacotes através da observação da janela de dados anunciada pelo lado receptor, ou seja, quantos bytes é possível inserir no campo de dados do protocolo TCP. Assim, o algoritmo opera através da verificação da taxa de inserção de pacotes na comunicação e a respectiva resposta de segmentos ACK do lado receptor. O algoritmo adiciona uma nova janela ao segmento TCP: O identificador de obstrução do meio denominado *Congestion Window(CWND).* Ao iniciar uma nova conexão, o CWND é inicializado com um único segmento (tamanho do segmento idêntico ao informado pelo lado receptor) . A cada novo ACK recebido, a janela CWND é dobrada.

O algoritmo negocia assim a máxima quantidade de pacotes que são enviados sem a necesssidade de espera por um pacote ACK de confirmação. O lado transmissor inicia o envio de um pacote e aguarda o ACK correspondende a ser enviado pelo lado receptor. Ao receber o pacote ACK, a janela CWND é incrementada de um para dois pacotes,

transmitindo assim dois pacotes. Após cada um desses pacotes ser confirmado por um pacote ACK, a janela CWND é incrementada para quatro, provendo um aumento exponencial à janela AWND. Após algumas trocas, o limite de comunicação do meio é alcançado.

O algoritmo *Slow Start* permite o ajuste da taxa de transmissão de pacotes numa rede onde há tráfego intenso através de roteadores que poderiam cancelar o envio dos pacotes ao seu destino. No caso específico da arquitetura paralela em uso, o algoritmo pode prejudicar o início da conexão pois não permite o acesso à máxima taxa de transferência de pacotes permitida pelo padrão *Fast Ethernet* após o estabelecimento de uma conexão.

3. Custos de Comunicação

Após o estabelecimento da conexão entre dois nodos da máquina paralela, a transferência de dados é processada.

Os dados são entregues pela aplicação à camada de transporte, são formatos devidamente pelo protocolo TCP que os repassa à camada de rede e os entrega para o envio ao lado receptor.

Os custos envolvidos nesse processo implicam também na transmissão e recepção no lado receptor, onde seguirão caminho inverso até serem entregues na camada de aplicação. Assim, o tempo gasto neste caminho determina o custo da comunicação. Desde o tempo gasto para a devida formatação dos dados até a camada de aplicação do lado receptor os entregar para o programa.

O custo de envio de uma mensagem entre dois nodos localizados em processadores diferentes pode ser representado de forma simplificada por dois parâmetros[4]:

- Tempo de inicialização da mensagem - t_s;
- Tempo de transferência da mensagem - t_w .

O tempo de inicialização é o tempo gasto para que o processo de comunicação se inicie. O tempo de transferência da mensagem determina o tempo de comunicação que conecta o remetente da mensagem ao destinatário. Esse tempo deve ser avaliado com o envio de uma palavra de quatro bytes que representa uma unidade mínima de dados, por exemplo, um inteiro. Assim, o custo de comunicação para o envio de uma mensagem com 'S' bytes pode ser estimado por:

$$T_{msg} = t_s + t_w S$$

A variação do parâmetro ´S´ permite estabelecer uma equação para o custo. Assim, ao plotar a equação de primeiro grau acima, o coeficiente angular fornece o custo unidade de inteiro transmitido e o coeficiente linear fornece o custo da inicialização, denominado latência. Assim, a latência é a forma de avaliar a velocidade com que os sistemas internos da arquitetura sincronizam dois processos remotos que cooperam na troca de mensagens. A figura 3.1 mostra o custo de comunicação

Utiliza-se a caracterização da largura de banda como medida da velocidade com que dados são transferidos numa conexão entre dois pontos remotos. A relação entre o total de bytes 'S' transmitidos e o tempo necessário 'T' para atingir seu destino denomina-se largura de banda, e é estimada por:

$$BW = S/T$$

O impacto da competição por largura de banda é maior nas aplicações que executam tarefas de forma síncrona, por exemplo, nos casos onde todos os processadores enviam e recebem mensagens simultaneamente e onde tarefas em diferentes processadores ficam ociosas à espera de dados para prosseguir com a sua computação.

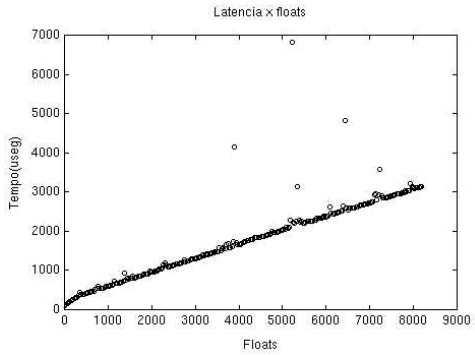

Figura. 3.1 – Custo de Comunicação

O custo da comunicação[6] na transferência de dados entre dois pontos pode ser estimado por:

$$T = L + S/BW$$

O parâmetro 'BW' representa a largura de banda e 'S' o total de *bytes* transferidos. A latência está representada pelo parâmetro ´L´ e o resultado ´T´ corresponde ao custo da comunicação.

O gráfico da taxa de transferência de bytes em função da dimensão da mensagem(T(S)) mostra a curva de custo de comunicação em função da quantidade de bytes transmitida. O gráfico de T(S) onde 'S' representa o comprimento da mensagem e 'D(S)' o tempo para envio da mensagem de comprimento 'S', é obtido por:

$$TC(S) = S / T(S)$$

Para valores grandes de ´S´, TC(S) representa o limite máximo da curva, a largura de banda. O gráfico apresentado na figura 3.1 mostra o custo de comunicação em função da variação do comprimento dos dados

enviados. O coeficiente linear desta reta indica a latência, para o caso em análise é de 300 microsegundos. O coeficiente angular fornece o custo adicional a cada elemento do tipo *float* enviado, o que corresponde a 330 microsegundos/float. Os pontos fora da reta representam a atuação do algoritmo de Nagle, onde os fragmentos pequenos são acumulados e assim aumentam o tempo de comunicação.

A figura 3.2 mostra a variação da taxa de transferência de dados em função da variação do comprimento dos dados enviados.

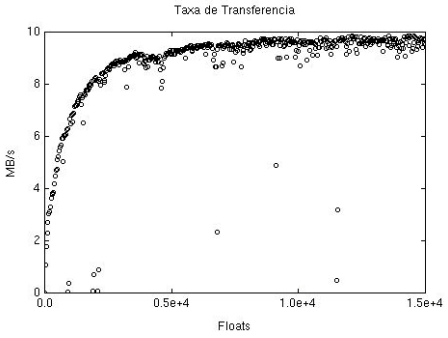

Figura 3.2 – Taxa de Transferência

Observa-se neste gráfico também a atuação do algoritmo de Nagle, representada no pontos traçados fora da curva apresentada. A taxa de transferência tende à largura máxima de banda para valores transmitidos altos, superiores a 5000 elementos do tipo *float*. A taxa de transferência mostrada corresponde a quatro conexões estabelecidas. algoritmo *Slow Start* pode ser observado no gráfico apresentado na figura 3.3.

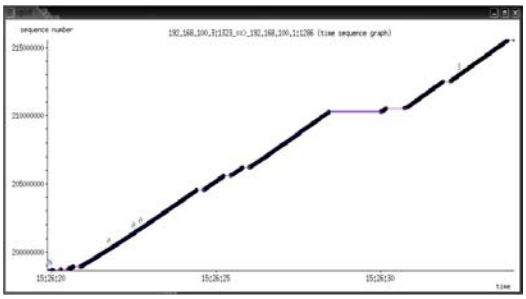

Figura 3.3 – Tráfego de Pacotes

Este gráfico apresenta a sequência de pacotes enviados em função do tempo. O traço mais escuro corresponde aos pacotes enviados. O traço mais fino representa o tempo de espera por segmento ACK. O início da transação entre os nodos mostra a negociação pelo algoritmo *Slow Start* do valor a ser utilizado pela janela CWND.

4. Conclusões

Com o objetivo de otimizar o desempenho e identificar a influência de parâmetros específicos, foi realizado uma análise dos custos de comunicação envolvidos na execução de programas paralelos utilizando a biblioteca MPI numa máquina paralela de memória distribuída.

Resultados preliminares mostram que o comutador pode manter a máxima taxa de transferência sem a ocorrência de colisões, ou seja, sem retardos de transmissão, sendo a largura de banda máxima utilizada simultaneamente por todos os 16 nodos da máquina paralela em análise .

A análise dos algoritmos específicos à implementação dos protocolos TCP/IP mostra que a utilização do algoritmo de Nagle penaliza o tempo de comunicação desnecessariamente, sendo conveniente sua desativação, o mesmo se aplica ao algoritmo *Slow Start*.

O protocolo TCP implica no estabelecimento de uma conexão e este e os demais protocolos implicam também no acréscimo e verificação de cabeçalhos, tudo isso onerando a comunicação. Assim, torna-se mais eficiente ter uma quantidade maior de mensagens com maior volume de dados.

No caso dos programas paralelos em questão, as dependências de dados entre processadores demandam comunicação por troca de mensagens, e a biblioteca MPI deve ser utilizada levando em conta essa característica de eficiência para oimizar a comunicação. Por exemplo deve-se minimizar sempre que possível o número de pares *send-recv*.

O tempo de comunicação pode ser dividido numa parte independente do tamanho da mensagem, a latência e outra parte associada à largura de banda da rede. O uso de um número menor de mensagens com volume maior de dados minimiza a influência da latência no tempo total de comunicação. Entretanto, é sempre desejável uma rede com maior largura de banda para minimizar o tempo de transferência associado ao tamanho da mensagem.

Finalmente, convém notar que os preceitos clássicos de otimização de códigos sequenciais continuam válidos nas implementações paralelas, uma vez que, obviamente, cada processador executa um programa sequencial nos seus próprios dados locais. Estes preceitos visam maximizar a utilização dos *pipelines* internos aos processadores e otimizar a utilização do sistema de memória, miimizando os acessos à memória principal ou à área de *swap*.

5. Referências

[1] Pacheco, S.P. *Parallel Programming with MPI***.** Morgan Kauffman Publishers, Inc. EUA, 1997.

[2] Stevens, W.R. *TCP/IP Illustraded, Volume 1 , The Protocols*. Addison-

Wesley, Inc. ,EUA, 1994.

[3] Minshall, G.; Saito, Y.; Mogul, J.; Verghese, B. “Application performance pitfalls and TCP's Nagle algorithm”**.** Workshop on internet performance. EUA, May 1999.

[4] Allman, M.; Floyd, S.; Partridge, C. “Increasing TCP's Initial Window”. RFC 2414. Network Information Center. September 1998.

[5] Stevens, W. “TCP Slow Start,Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithms”**.** RFC2001. Network Information Center. 1997.

[6] Buyya, Rajkumar. *High Performance Cluster Computing - Architectures and Systems*. Prentice Hall PTR. EUA, 1999