Troca de mensagens através de memória compartilhada para o ambiente DECK

Caciano Machado*, Rafael B. Ávila†,
Philippe O. A. Navaux‡

Grupo de Processamento Paralelo e Distribuído — GPPD
Universidade Federal do Rio Grande do Sul — Instituto de Informática
{caciano,avila,navaux}@inf.ufrgs.br

Introdução e motivação

O modelo de programação paralela baseado em troca de mensagens é utilizado como padrão em multicomputadores de todas as classes. No caso de agregados de estações de trabalho (*clusters of workstations*) [BUY 99] essa comunicação se faz através de uma rede de interconexão como Ethernet ou tecnologias específicas e mais rápidas como Myrinet e SCI com ajuda de bibliotecas com suporte a troca de mensagens como MPI [MPI 94] e DECK [BAR 2000]. Considerando o caso específico de agregados compostos por nodos SMP , abre-se a possibilidade de comunicação através da memória endereçável pelos processadores da máquina. O presente trabalho visa a utilização da memória compartilhada entre processos sendo executados em um mesmo nodo como repositório intermediário para as mensagens permutadas entre os processos. A implementação do mecanismo consiste na criação de novas primitivas para armazenamento e recuperação da mensagem em um segmento de memória alocado. Parte deste trabalho foi publicada no PDPTA-2002 [ÁVI 2002] com suporte para comunicação apenas entre *threads* de um mesmo nodo do agregado.

A biblioteca DECK

DECK—*Distributed Execution and Communication Kernel* é uma biblioteca de programação paralela que provê os recursos considerados fundamentais para a execução de aplicações em agregados de alto desempenho.

A API da biblioteca provê as abstrações mais básicas para o desenvolvimento de aplicações paralelas para *clusters*: *threads*, *mutexes*, semáforos, variáveis de condição, *mail boxes* e mensagens. As primitivas correspondentes a estas abstrações formam a camada mais interna da biblioteca denominada μDECK. Além disso DECK disponibiiza um conjunto de serviços que utilizam as primitivas do μDECK.

A comunicação entre os fluxos de execução da aplicação (*threads* e/ou processos) é feita através de *mail boxes* com o uso de troca de mensagens. Cada processo pode

*Graduando em Ciência da Computação, Bolsista de IC
†Doutorando em Ciência da Computação, Co-Orientador
‡Professor, Doutor (Institut National Polytechnique de Grenoble, França, 1979), Orientador

criar uma ou mais *mail boxes* onde receberá as mensagens. Caso haja necessidade de comunicação, a *thread* ou processo remetente deve requisitar informações sobre a *mail box* destino (*deck_mbox_clone*) para que possa enviar mensagens.

O envio da mensagem (*deck_mbox_post*) é feito de maneira assíncrona (não-bloqueante) e o recebimento (*deck_mbox_retrv*) é feito de maneira síncrona (bloqueante).

Os protocolos de comunicação para troca de mensagens disponíveis atualmente são TCP/IP (redes Ethernet), GM (redes Myrinet) e SISCI (redes SCI), além do protocolo de comunicação por memória compartilhada que será descrito.

Troca de mensagens através de memória compartilhada

Os segmentos de memória utilizados para a comunicação entre os processos foram alocados utilizando as rotinas IPC System V disponíveis em diversos sistemas UNIX. Cada processo aloca um segmento de memória de 4 MB onde serão recebidas as mensagens de cada *mail box* criada neste processo. Além disso cada segmento é anexado a todos os outros processos que estão sendo executados no mesmo nodo do *cluster* para que estes possam enviar mensagens ao o processo criador do segmento.

A primitiva *deck_shmem_post* responsável pelo envio de mensagens consulta uma tabela contendo os identificadores dos processos e de seus respectivos segmentos de memória para encontrar o segmento de destino da mensagem. A primitiva *deck_shmem_retrv* que executa o recebimento das mensagens pesquisa o segmento criado pelo processo à procura de mensagens recebidas.

A criação de *mail boxes* é feita por demanda, ou seja, quando a comunicação através da memória for possível é criada uma *mail box* no segmento de memória que irá receber as mensagens.

A tranferência de mensagens pequenas (até 8KB) é feita com cópia do descritor e buffer da mensagem em um segmento de 8KB. Para mensagens longas (maiores que 8KB) é feita a segmentação em pacotes de 8KB.

O controle de acesso aos segmentos e às suas variáveis é feito utilizando um mecanismo de exclusão mútua menos oneroso se comparado com os do IPC System V e Pthreads, a função *test_and_set_bit* fornecida pelo Kernel do Linux. O endereço inicial do segmento de memória possui o *lock* que controla o acesso à tabela de *mail boxes*. Além disso cada *mail box* possui um *lock* que bloqueia seu acesso depois que ela foi encontrada (ou criada) na tabela de *mail boxes*. Isso permite que várias mensagens sejam enviadas para *mail boxes* diferentes de um mesmo segmento de memória e que a tabela possa ser consultada ao mesmo tempo.

O mecanismo de troca de mensagens implementado é escondido na API do DECK de forma que seja possível utilizar o protocolo padrão e o de memória compartilhada de maneira transparente de acordo com as necessidades de comunicação.

Avaliação de desempenho

Foi feita uma comparação dos valores de latência e largura de banda de mecanismos de troca de mensagens por memória com os de troca de mensagens utilizando o

protocolo TCP/IP num mesmo nodo do *cluster*.

O teste realizado foi um *ping-pong* entre dois processos cada qual sendo executado em um processador da máquina biprocessada. Foi feita a média de 1000 execuções do teste para evitar casos isolados.

Para realização dos testes foi utilizado um nodo SMP Dual Pentium III 500 MHz com 256 MB de memória RAM. O software utilizado foi o sistema operacional Linux com Kernel 2.4.12 e Glibc 2.2.3 e as bibliotecas de programação paralela DECK 2.2.1 e MPICH 1.2.4a.

No DECK executamos os testes utilizando a versão de *mail boxes* para TCP/IP e outra com as rotinas de comunicação por memória.

Para o MPICH [PRO 2002] foram feitos testes com o dispositivo *ch_shmem* e *ch_p4*. Não foram realizados testes com o dispositivo *ch_lfshmem* por que atualmente só existe implementação para a arquitetura NEC SX-4. O dispositivo *ch_shmem* utiliza comunicação por memória compartilhada em nodos SMP. O mecanismo de troca de mensagens deste dispositivo utiliza dois protocolos distintos dependendo do tamanho da mensagem. As mensagens pequenas são enviadas para áreas de memória pré-alocadas e as longas são armazenadas em regiões de memória alocadas dinamicamente. O dispositivo *ch_lfshmem* além disso utiliza um mecanismo de sincronização *lock-free* que minimiza o overhead de sincronização.

Observou-se um desempenho significativamente superior do protocolo desenvolvido sobre os demais mecanismos de comunicação para valores de mensagem de até 64KB. Acima disto os valores de latência e largura de banda dos diferentes mecanismos se aproximam pois o overhead de comunicação se torna menos expressivo em relação ao tamanho das mensagens.

Os valores de latência obtidos para mensagens pequenas atingem aproximadamente 4.3 μs e a largura de banda alcança picos de cerca de 140 MBytes na máquina testada.

Nota-se uma ligeira vantagem do mecanismo *ch_shmem* do MPICH para mensagens acima de 64KB. Essa vantagem se deve ao fato que o tratamento da comunicação para mensagens longas do MPICH utiliza alocação dinâmica de memória e a do DECK faz a segmentação da mensagem em pacotes de 8KB. Isso faz com que o uso da memória *cache* do processador fique prejudicada já que os pedaços da mensagem poderão estar fisicamente muito distantes na memória pricipal.

Conclusões e trabalhos futuros

Explorando intensivamente os recursos de cada nodo independentemente as aplicações paralelas poderão tirar o melhor proveito do agregado como um todo.

Os valores obtidos na avaliação de desempenho demonstram a importância deste tipo de comunicação para exploração plena dos recursos de agregados de multiprocessadores. Foi alcançado um aumento de desempenho na comunicação intra-nodo de cerca de 1200% para latência e mais de 400% para largura de banda com relação a comunicação utilizando TCP/IP.

Ainda restam algumas atividades que devem ser finalizadas no mecanismo de troca de mensagens por memória. Entre estas atividades as principais estão a otimização do

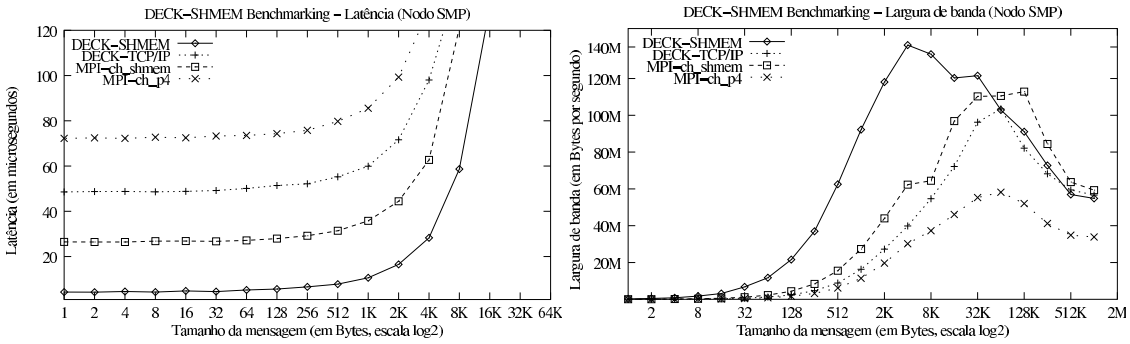

Figura 1: Testes de Latência e Largura de Banda

algoritmo de ordenamento de mensagens no caso de *mail boxes* que recebam mensagens de processos locais e remotos e melhora do protocolo para alcançar níveis ótimos de latência e largura de banda. Além disso está sendo estudada a possibilidade de criação de um algoritmo *lock-free* para o protocolo.

Agradecimentos

Nossos agradecimentos para CNPq e Dell, financiadores das bolsas e equipamentos que permitiram o desenvolvimento destas pesquisas.

Referências

[ÁVI 2002] ÁVILA, R. B.; MACHADO, C.; NAVAUX, P. O. A. Message-passing over shared memory for the DECK programming environment. In: INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED PROCESSING TECHNIQUES AND APPLICATIONS, 2002, Las Vegas. **Proceedings...** Las Vegas: CSREA Press, 2002. p.591–595.

[BAR 2000] BARRETO, M. E. **DECK**: um ambiente para programação paralela em agregados de multiprocessadores. 2000. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

[BUY 99] BUYYA, R. (Ed.). **High performance cluster computing**: architectures and systems. Upper Saddle River: Prentice Hall PTR, 1999. 849p.

[MPI 94] MPI FORUM. **The MPI message passing interface standard**. Knoxville: University of Tennessee, 1994.

[PRO 2002] PROTOPOPOV, B. V. **Comparison of designs of shared memory devices for mpich**. Available at http://www.cs.msstate.edu/∼boris/papers/ShmemDev.ps.