Modelagem da Comunicação no Ambiente DECK Utilizando GM para Myrinet[1]

Clarissa C. Marquezan, Rafael B. Ávila, Philippe O. A. Navaux

Instituto de Informática - Universidade Federal do Rio Grande do Sul - UFRGS
Av. Bento Gonçalves, 9500 - Porto Alegre RS - Telefone (51)3316 6846 - Fax: (51)3316 1576
{clarissa, avila, navaux}@inf.ufrgs.br

Introdução

Através do uso de agregados de computadores a cultura do processamento de alto desempenho vem se disseminando cada vez mais. O crescimento do uso de clusters assim como o aumento do número de nodos que os compõem leva a necessidade da utilização de equipamentos de interconexão mais rápidos. Como a comunicação entre os nodos é mais onerosa, se comparada com o processamento em cada nodo, o uso de uma rede de comunicação com melhor desempenho permite atingir um aproveitamento maior dos recursos disponíveis. Um bom exemplo disso é o emprego de Myrinet [BOD 95] em um cluster. Através dessa tecnologia pode-se obter taxas de transmissão na ordem de 2Gb/s para cada canal (full duplex), melhorando o desempenho do cluster.

Esse trabalho tem como objetivo apresentar a implementação do ambiente DECK [OLI 01], [BAR 00] utilizando Myrinet. Para tanto foi realizado um estudo utilizando a biblioteca GM [MYR 02], desenvolvida pela Myricom no intuito de servir como uma camada entre aplicativos ou ambientes (DECK/MPI) e a interface Myrinet.

Biblioteca GM

A biblioteca GM foi desenvolvida por pesquisadores da Myricom, é uma biblioteca para troca de mensagens baseada em eventos e em reserva de tokens. As características fundamentais do GM são: baixa latência; alta banda passante; entrega confiável e ordenada dos pacotes; *connectionless reliability*, a multiplexação do fluxo de dados, assim como a manutenção das ligações entre as portas garantem essa característica; utilização de *DMAable memory*; apresenta dois níveis de prioridades (baixa e alta); mensagens e buffers são identificados pela prioridade, *size* e *length.*

Uma mensagem só é recebida corretamente se existir um buffer de recebimento com o mesmo *size* e prioridade da mensagem, caso contrário um evento de *timeout* é gerado. Outra característica fundamental para comunicação através da biblioteca GM é a alocação de *DMAable memor*y tanto para o envio como para o recebimento de mensagens, caso isso não seja providenciado não há garantia de corretude dos dados. Associado às características já descritas acima, a alocação de *tokens* de recebimento e envio de mensagens fecha os requisitos básicos para que uma comunicação através da biblioteca GM seja efetuada.

[1]Trabalho parcialmente financiado pelo convêncio PIBIC/CNPq, FINEP e LabTeC - UFRGS/DELL.

Ambiente DECK

O DECK (Distributed Execution and Communication Kernel) é um ambiente voltado para a computação de alto desempenho. Através de sua API [MAC 02] o usuário tem primitivas de programação distribuída e de programação por variáveis compartilhadas (*multithreading*). É um ambiente modular, composto pelo µDECK no qual estão as funcionalidades básicas do ambiente e um módulo de serviço. Alterações no µDECK possibilitam a adaptação desse ambiente para diferentes tipo de tecnologias de interconexão, sem modificações nos aplicativos que o utilizam.

Atualmente o DECK está disponível em três versões: Fast Ethernet, SCI, BIP para Myrinet. Devido à descontinuidade do desenvolvimento da biblioteca BIP e também por problemas de controle de fluxo no desenvolvimento do DECK baseado nessa biblioteca (o BIP não trata o fluxo das mensagens, fica a cargo dos usuários) decidiu-se portar o ambiente DECK para a biblioteca GM. Dispondo, assim, dos recursos de alto desempenho que uma rede baseada em Myrinet propicia.

Modelo de Comunicação Proposto

A maior dificuldade para implementar o DECK sobre o GM foi adaptá-lo ao paradigma de eventos. A cada envio ou recebimento de mensagens um evento é gerado e deve ser tratado. O problema é fazer com que as primitivas de comunicação do DECK recebam esses eventos sem interfir em suas semânticas. A solução encontrada foi o uso de uma *thread* que centraliza todos os eventos. Essa *thread*, denominada de *ReceiverThread*, é criada durante a inicialização do ambiente DECK.

Devido à necessidade da existência de buffers e tokens previamente alocados para cada *size* de mensagens que se deseja receber, foi necessário que, também na inicialização do DECK, fossem alocados buffers e tokens. Mensagens pequenas no GM (até 120 bytes) não necessitam de *DMA*, são mantidas dentro da interface Myrinet, no entanto mensagens maiores precisam ser alocadas e recebidas em *DMAable memory*.

Através de testes percebeu-se um limite na quantidade de *DMA* que podia ser alocada no cluster onde os testes foram realizados (LabTeC - UFRGS/DELL). Para resolver o problema da quantidade de memória *DMA* alocável inicialmente pelo DECK dividiu-se a classe de mensagens grandes do GM em duas: mensagens médias (com buffers alocados inicialmente) e mensagens grandes (com buffers alocados durante a execução). No presente trabalho serão apresentadas as soluções para envio e recebimento de mensagens pequenas e médias, utilizando apenas a prioridade baixa fornecida pelo GM. A Figura 1 mostra o processo de comunicação no ambiente DECK.

Enviando Mensagens

O processo de envio de dados é mais simples se comparado com o modelo de recebimento. Basicamente um *token* para envio é reservado, o buffer que será enviado é alocado em uma DMAable memory e a mensagem é enviada. A cada envio um evento é retornado, nele pode-se saber o status da transmissão (sucesso ou erro). Esse evento é capturado pela primitiva *gm_unknown()* dentro da *ReceiverThread*. Essa primitiva é

responsável por ativar a chamada de uma função de retorno, denominada *CallBack()*. Através dela tem-se acesso ao tipo de evento que ocorreu durante a transmissão da mensagem. Em caso de erro, o GM disponibiliza primitivas para restransmissão dos dados (procedimento ainda não implementado nessa versão do DECK-GM).

Recebendo Mensagens

Durante a inicialização do DECK uma fila de *Mail Boxes* é criada. A utilização dessa estrutura foi necessária para que a *ReceiverThread* pudesse entregar corretamente as mensagens recebidas. A cada posição dessa fila está associada uma *mail box* e um buffer de armazenamento de mensagens. Sendo assim, uma *mail box* pode receber mais de uma mensagem sem que a primeira tenha sido retirada desse buffer.

Assim que um evento de recebimento é reconhecido pela *ReceiverThread* os dados são transferidos para a posição da fila referente à *mail box* destino. Caso seja uma mensagem pequena ela será copiada da própria estrutura do evento para esse buffer, caso seja de tamanho médio será copiada do buffer alocado com *DMA*.

Quando um *deck_mbox_retrv()* é chamado ele fica bloqueado esperando que uma mensagem seja armazenada no buffer da fila de *Mail Boxes*. Depois de desbloqueado, a mensagem recebida é novamente copiada para a estrutura de mensagens do DECK. Essas cópias sucessivas foram necessárias devido a dois motivos. Primeiro, o espaço de *DMA* que pode ser alocado é pequeno e da forma como foram definidas as estruturas para a comunicação ela pode ser utilizada por mensagens destinadas a diferentes *mail box*. Segundo, a semântica do DECK não associa os buffers de recebimento às *mail boxes* e sim às estruturas de mensagens DECK (*deck_msg_t*).

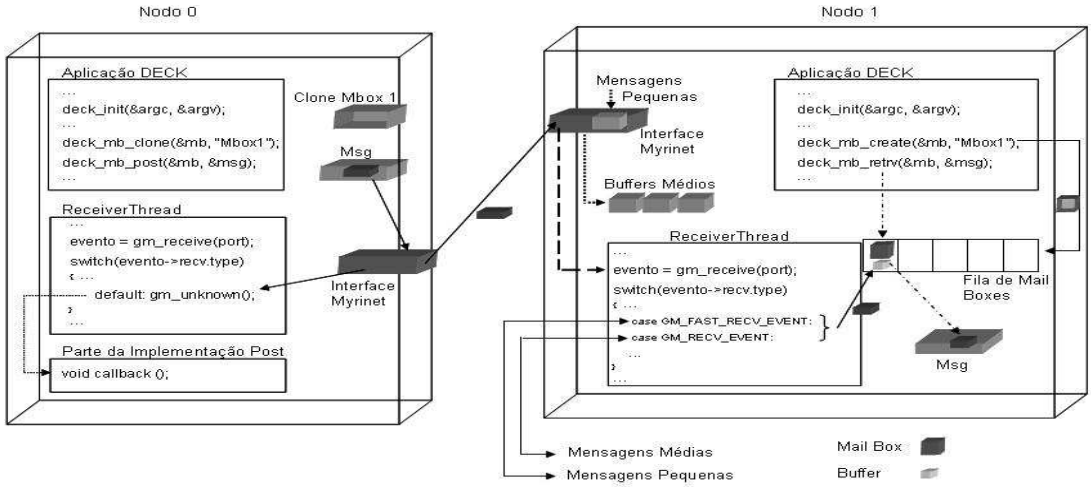

Figura 1 - Modelo de comunicação do DECK-GM

Medidas Alcançadas

Para aquisição dos resultados utilizou-se o cluster LabTeC - UFRGS/DELL instalado no Instituto de Informática. As máquinas utilizadas foram dois Pentium III biprocessados com 1,13 GHz e 1 GB de memória RAM. Para as medidas foram utilizados apenas um processador de cada nodo. A aplicação que serviu como base de

comparação foi um *pingpong,* implementado em DECK e MPI. Também foi utilizada a aplicação *gm_all_size()* disponibilizada na distribuição do GM para medir sua latência.

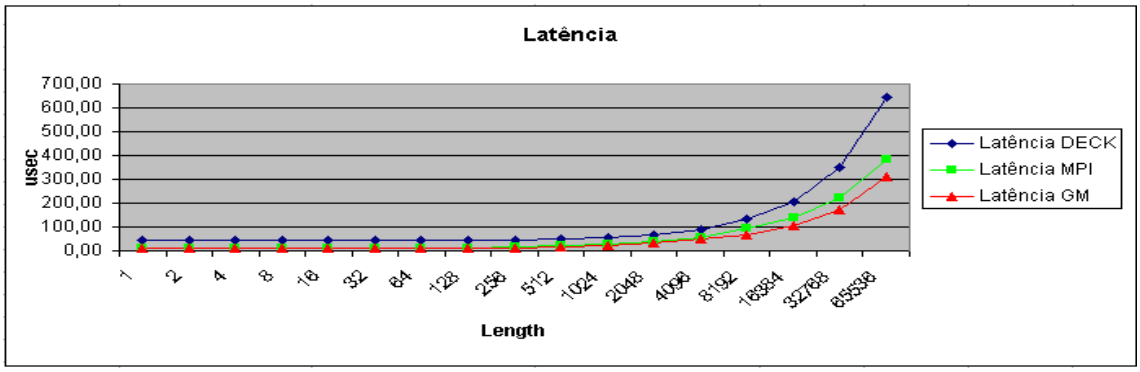

Figura 2 - Comparação de latência

Através dos resultados obtidos na Figura 2 percebeu-se que as sucessivas cópias para os buffers internos do DECK se mostram prejudiciais principalmente na transmissão de mensagens pequenas, se comparado com as latências obtidas com o MPI-GM e o GM. A forma mais direta de melhorar o desempenho do ambiente DECK é diminuir a quantidade de cópias das mensagens recebidas, visando *zero-copy*.

Conclusões e Trabalhos Futuros

A implementação do DECK sobre o GM possibilitou uma melhor utilização dos recursos do cluster LabTeC - UFRGS/DELL, e que o ambiente DECK possa ser utilizado com mais uma tecnologia de interconexão, tornando-o mais portável.

Como trabalhos futuros estão a implementação da transmissão de mensagens grandes, assim como sua segmentação, implementação de um mecanismo de retransmissão em caso de erro, definição do limiar entre mensagens médias e grandes propiciando melhor desempenho e implementação de multiplexação de porta dos nodos.

Referências

[BAR 00] BARRETO, M; ÁVILA, R.; CASSALI, R; CARISSIMI, A.; NAVAUX, P. **Implementation of the DECK Environment with BIP.** MUG 2000, p. 82-88.

[BOD 95] BODEN, N. et al. **Myrinet**: A gigabit-per-second local-area network. **IEEE Micro**, Los Alamitos, v.15, n.1, p.29--36, Feb. 1995.

[MAC 02] MACHADO, C.; MARQUEZAN C. **Tutorial on DECK - Distributed Execution and Communication Kernel.** Tutorial desenvolvido e distribuído com o ambiente DECK.

[MYR 02] MYRINET. **Myrinet:** Software & Documentation. Disponível por www em http://www.myri.com/scs/ (10/ 11/ 2002).

[OLI 01] OLIVEIRA, F.; ÁVILA, R; BARRETO. M; Navaux, P.; DeRose, C. **DECK-SCI:** High-Performance Communication and Multithreading for SCI Clusters. Cluster 2001, p. 372-379.