libVIP: Arquitetura de Interface Virtual (VIA) sobre TCP/IP*

Rodrigo da Rosa Righi, Philippe O. A. Navaux
PPGC - Prog. de Pós-Graduação em Computação
Universidade Federal do Rio Grande do Sul
{rrrighi,navaux}@inf.ufrgs.br

Marcelo Pasin
Laboratório de Sistemas de Computação
Universidade Federal de Santa Maria
pasin@inf.ufsm.br

Resumo

Aglomerados de computadores são freqüentemente utilizados como arquitetura de suporte ao processamento de alto desempenho. Os computadores nesta arquitetura se comunicam através de trocas de mensagens e existe atualmente pesquisa para torná-las mais eficientes. Em 1997, foi especificada a Arquitetura de Interface Virtual (VIA) [5]. VIA é uma interface de programação que possibilita a implementação eficiente de bibliotecas de comunicação. Baseada na arquitetura VIA, foi construída a biblioteca para comunicação assíncrona libVIP [16]. Esta biblioteca foi implementada usando a camada de rede padrão de sistemas POSIX. Ela possibilita a execução de programas paralelos escritos para a interface VIA em quaisquer aglomerados de computadores, mesmo naqueles que não possuem interfaces de rede que seguem este padrão. O presente artigo tem por objetivo apresentar a implementação da libVIP e descrever duas aplicações desenvolvidas para avaliá-la.

1. Introdução

À medida que evoluem as tecnologias envolvendo a construção de computadores, também cresce a quantidade de aplicações que necessitam de grande poder de processamento. Aglomerados de computadores (*clusters*), montados a partir de computadores comuns interligados por uma rede local rápida, vêm sendo usados em atividades comerciais e acadêmicas como sistemas capazes de fornecer computação de alto desempenho [3]. O sucesso dessa arquitetura deve-se às vantagens que ela apresenta com relação aos supercomputadores tradicionais, tais como mais baixo custo por unidade de desempenho, mais flexibilidade e mais escalabilidade [3, 17].

Com o crescimento de aplicações que se servem de aglomerados de computadores, tem-se buscado maneiras para melhorar o seu desempenho. Uma destas maneiras é o aumento da eficiência da comunicação entre os computadores de um aglomerado, chamados **nós**. Para maximizar o desempenho computacional em aglomerados de computadores, foi publicado, em 1997, a especificação da Arquitetura de Interface Virtual, ou VIA [5], que é uma interface de programação que visa permitir a implementação eficiente de mecanismos de comunicação em redes de sistema, aumentando a vazão de dados, reduzindo a carga de processamento local e diminuindo o tempo para a troca de mensagens entre os nós de uma rede [7].

A arquitetura VIA define comportamentos específicos para uma interface de rede e seu módulo de controle no sistema operacional. As bibliotecas que implementam a interface VIA são restritas a alguns adaptadores de rede específicos e geralmente são de difícil instalação, passando necessariamente pela adição de módulos ao núcleo do sistema operacional [12]. Isto colabora para aumentar a complexidade de depuração do código-fonte e dificulta a portabilidade da biblioteca, pois cada adaptador de rede necessita o seu próprio módulo.

Baseado nesse contexto, foi desenvolvida a biblioteca de comunicação assíncrona VIP (*Virtual Interface Protocol*), ou libVIP, adaptando-se a arquitetura VIA para fornecer alta portabilidade. O presente artigo possui a finalidade de apresentar a implementação dessa biblioteca e as tecnologias utilizadas para a sua construção. O artigo também apresenta a análise de duas aplicações escritas para avaliar a libVIP.

A biblioteca VIP foi implementada no sistema operacional Linux, usando *sockets* [4] e protocolo de comunicação TCP/IP (*Transmission Control Protocol/Internet Protocol*) [4, 18, 14]. Nenhuma funcionalidade particular ao Linux foi utilizada, sendo assim, a biblioteca teoricamente portável para qualquer sistema do tipo POSIX [13]. A libVIP possibilita escrever aplicações com a arquitetura VIA para quaisquer computadores interligados com o protocolo de rede tradicional da Internet, sem a necessidade de instalação de dispositivos e módulos de rede específicos para VIA. Havendo disponibilidade de adaptadores de rede para VIA, poder-se-á reutilizar os mesmos programas desenvolvidos com a libVIP. Além disso, a libVIP é uma biblioteca sim-

*Financiamento CNPq, processos 181589/01-8 e 380049/03-1

ples e de fácil depuração, visto que foi implementada em nível de endereçamento de memória do usuário.

2. A Arquitetura de Interface Virtual - VIA

Um dos fatores mais importantes para obter desempenho na transferência de mensagens é a redução da sobrecarga de *software*. A sobrecarga de *software* é proveniente de muitas fontes [6], como a complexidade da implementação do protocolo de rede, ativação de interrupções, trocas de contexto, mecanismos de roteamento, cópia de mensagens, etc. Para minimizar a sobrecarga de *software* e fornecer um rápido caminho entre as aplicações e a placa de rede, são propostos métodos para remover o núcleo do sistema operacional e estruturas de dados intermediárias dos protocolos de comunicação entre computadores.

Com o intuito de padronizar um conjunto de protocolos de comunicação de alto desempenho (*U-Net, Active Messages, Fast Messages*) [12], em 1997 foi desenvolvido pelas empresas Intel, Compaq e Microsoft a especificação da Arquitetura de Interface Virtual, ou VIA. Esta arquitetura provê para cada processo cliente um acesso protegido e direto à interface de rede, de onde o nome interface virtual [5]. Ela fornece baixa latência de comunicação através da redução das trocas de contexto (entre o espaço do usuário e o núcleo do sistema operacional) nas rotinas para transferência de mensagens e da minimização das cópias intermediárias de memória [2, 7, 8]. O núcleo do sistema operacional é envolvido somente durante a conexão, quando este realiza a verificação das permissões necessárias. Depois de realizada a conexão, a transmissão dos dados ocorre diretamente entre a área de memória do usuário e a placa de rede VIA. O adaptador de rede VIA realiza as tarefas de multiplexação, demultiplexação e agendamento de transferência de dados, que geralmente são questões inerentes ao sistema operacional e a um *driver* de dispositivo [5].

A arquitetura de Interface Virtual é composta de quatro componentes: Interfaces Virtuais (VIs), fila de requisições completas, Provedor de Interface Virtual e Cliente de Interface Virtual. Esta arquitetura está ilustrada na figura 1. O Provedor VI é composto por um adaptador de rede e uma parte de *software*, chamada de agente do núcleo. O Cliente VI é geralmente composto de um programa de aplicação e alguma biblioteca de comunicação.

A figura 2 apresenta um esquema de como funciona a transferência de dados com a Arquitetura de Interface Virtual. O agente do núcleo, que está no sistema operacional, controla a interface do adaptador de rede e gerencia as aplicações para acessá-la. As aplicações alocam regiões de memória em seus espaços de endereçamento e chamam o agente do núcleo para estabelecer o acesso com a placa de rede. Uma vez realizado este procedimento, as aplicações podem iniciar a transmissão e recepção direta de dados de e

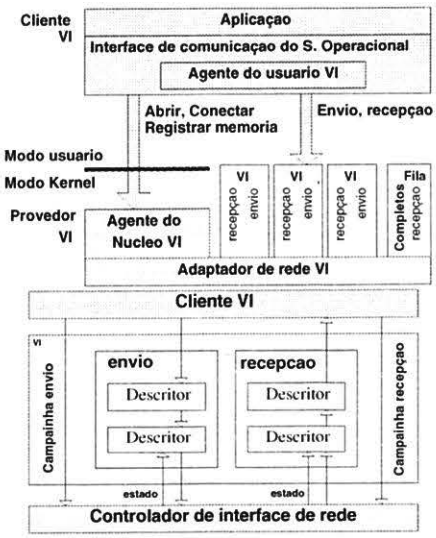

Figura 1. Componentes da Arquitetura de Interface Virtual

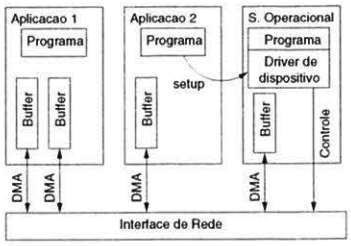

Figura 2. Transferência de Dados entre as Aplicações e a Interface de Rede

para a região de memória utilizando o mecanismo de DMA (Acesso Direto a Memória) [7].

Na arquitetura VIA, uma interface virtual é análoga a um *socket* para conexão tradicional TCP: cada VI suporta transferência de dados bi-direcional e ponto-a-ponto [2]. Cada interface virtual permite a um Cliente VI acessar diretamente um Provedor VI para realizar operações de transferência de dados. Uma interface virtual consiste de um par de filas de trabalho: uma fila para envio e outra para recepcão. Clientes VI colocam suas requisições em forma de descritores nas filas de envio e de recepção de dados. Um descritor é uma estrutura de dados que contém todas as informações que o Provedor VI necessita para processar a requisição e também ponteiros para a área onde estão os dados. O Provedor VI processa assincronamente os descritores contidos nas filas, e marca seu respectivo valor de controle como completado [2, 5]. Os Clientes VI removem os descritores que tiveram suas operações finalizadas da fila de trabalho e estes poderão ser usados em uma futura

requisição.

Tabela 1. Principais Primitivas VIA para a Troca de Mensagens

Primitiva	Significado
`VipPostSend`	Adiciona um descritor na fila de envio de uma VI
`VipPostRecv`	Adiciona um descritor na fila de recepção de uma VI
`VipSendWait`	Espera por um descritor completo na fila de envio
`VipRecvWait`	Espera por um descritor completo na fila de recepção

A tabela 1 apresenta as principais funções da arquitetura VIA referentes à troca de mensagens. As duas primeiras funções possuem o objetivo de adicionar um descritor em uma das filas de uma interface virtual. As outras duas servem para esperar até que o primeiro descritor na fila de envio, ou recepção, tenha a sua requisição de troca de mensagem completada. Este desmembramento das funções de envio e recepção em duas partes permite a escrita de programas que utilizem o processador enquanto a comunicação está em andamento, aumentando a sua eficiência.

3. Implementação da Biblioteca VIP

Esta seção descreve as estruturas de dados e algoritmos adotados para implementar a libVIP em linguagem de programação ANSI C [9]. A biblioteca VIP implementa um subconjunto das principais diretivas de transferência de mensagens oferecidas por VIA. Ela foi desenvolvida com o sistema operacional Linux e faz uso das bibliotecas *Pthreads* [10], para o controle de fluxos de execução (*threads*), e de *sockets* TCP [4, 14], para realizar a comunicação entre computadores. Estas questões de projeto foram adotadas para fornecer os principais objetivos da libVIP, que são a portabilidade de *hardware* e a segurança na comunicação entre duas interfaces virtuais.

O protocolo de transporte TCP, apesar de impor uma sobrecarga na operação de troca de mensagens, proporciona alta portabilidade de *hardware*, sendo possível executar a libVIP em quaisquer adaptadores de rede configurados para operar com este protocolo. Para realizar uma comunicação, o procedimento de envio, ou recepção, de dados descrito na interface libVIP chama um procedimento da biblioteca de *sockets*, que por sua vez, realiza uma chamada de sistema e o sistema operacional realiza a interação com a interface de rede.

3.1. Fluxos de Execução

A arquitetura VIA realiza as operações de envio e recepção de mensagens de modo assíncrono. Esse assincronismo é proporcionado pela criação de fluxos de execução e filas de descritores. Os procedimentos de troca de mensagens, `VipPostSend` ou `VipPostRecv`, possuem exclusivamente a funcionalidade de colocar um descritor na fila de uma interface virtual e retornam imediatamente.

No momento que uma interface virtual é criada, são instanciados dois fluxos de execução. O primeiro fluxo de execução trata do envio, e o outro da recepção de dados de uma interface virtual. Ambos os fluxos de execução recebem como argumento a interface virtual criada. O fluxo de execução de envio funciona como um demônio (*daemon*, servidor) [10], que espera até que um descritor não completado seja retirado da fila da interface virtual, processa a operação de envio dos dados apontados por este descritor, marca este descritor como completado e retorna para retirar o próximo.

O par de procedimentos `VipPostSend` e o demônio de transferência constituem um par produtor-consumidor [15], onde o primeiro adiciona uma requisição através de um descritor na fila de envio e o demônio de envio possui a função de capturar o primeiro descritor incompleto e realizar a troca de mensagem. O funcionamento do fluxo de execução de recepção é análogo. Os fluxos de execução referentes aos demônios de envio e recepção de dados são terminados quando a conexão entre as interfaces virtuais local e remota é desfeita.

3.2. Controle de Concorrência

A necessidade de utilização de técnicas de exclusão mútua na libVIP pode ser entendida como segue. Na execução de uma aplicação, pode vir a existir um cenário onde existam dois fluxos de execução criados pelo usuário, e ambos executando o mesmo procedimento de envio ou recepção. Neste procedimento, a região de código que opera sobre a fila é uma região crítica, pois esta fila é uma variável compartilhada e ambos os fluxos de execução criados pelo usuário estão fazendo acesso a ela. Além disso, existem sempre dois fluxos de execução para cada interface virtual executando concorrentemente com as chamadas da biblioteca, que são os demônios de envio e de recepção. Estes dois fluxos de execução realizam acesso concorrente com os procedimentos de postagem de descritores nas filas de uma interface virtual e podem acontecer problemas de coerência e integridade dos dados. Para impedir estes eventuais problemas, são utilizadas duas variáveis de exclusão mútua. A primeira variável garante a exclusão mútua para regiões de código que operam sobre a fila de envio de uma interface virtual. O funcionamento da outra variável é análogo para a

fila de recepção.

O algoritmo de demônio de envio, ou recepção, procura por um descritor incompleto em uma das listas de uma interface virtual. Caso a lista esteja vazia ou não contenha descritores incompletos, este fluxo de execução deve esperar, sem ocupar ciclos da CPU, até que um descritor novo seja colocado na fila e uma notificação seja lançada. Para garantir essa funcionalidade, a libVIP utiliza o mecanismo de variáveis de condição [10, 15].

A biblioteca VIP disponibiliza, além das funções de postagem de descritores em uma das filas de uma interface virtual, duas funções que esperam até que um descritor tenha a sua requisição completada (tabela 1). Enquanto a requisição do descritor não estiver finalizada, ambos os procedimentos esperam, também sem ocupar a CPU, até que um descritor completado esteja disponível. O controle de espera e notificação para estes dois procedimentos também são tratatos por variáveis de condição.

3.3. Interface da Biblioteca VIP

A especificação da Arquitetura de Interface Virtual apresenta as características e o funcionamento deste protocolo de comunicação. Esta especificação também fornece as estruturas de dados que compõem a arquitetura e descreve o conjunto de procedimentos que VIA disponibiliza para o usuário. A libVIP implementa um subconjunto de 12 funções da especificação VIA 1.0 [5]. Estes procedimentos foram implementados seguindo sempre a mesma sintaxe e semântica descritos na especificação.

Tabela 2. Procedimentos Implementados pela libVIP

Conjunto	Procedimentos
Criação/destruição de uma VI	`VipCreateVi` `VipDestroyVi`
Gerenciamento da conexão	`VipConnectAccept` `VipDisconnect` `VipConnectRequest`
Transferência de dados	`VipPostSend` `VipPostRecv` `VipSendWait` `VipRecvWait` `VipSendDone` `VipRecvDone`
Resolução de nomes	`VipNSGetHostByName`

A tabela 2 apresenta os procedimentos implementados pela libVIP. A sintaxe e a semântica destes procedimentos e dos demais disponibilizados por VIA podem ser encontrados na especificação VIA 1.0 [5]. Com este conjunto de procedimentos, poder-se-á construir aplicações paralelas para aglomerados de computadores explorando as características da arquitetura VIA.

3.4. Modalidades de Comunicação

Quando uma interface virtual é criada, a função `VipCreateVi` recebe como parâmetro de entrada as características do ponto final de comunicação criado. Baseado nessas características, a libVIP disponibiliza duas modalidades de comunicação, o modo confiável e o modo não confiável. Caso o usuário especificar a modalidade de comunicação não confiável, são feitas algumas modificações no *socket* que representa o descritor de comunicação entre as duas VIs. Dessa forma, habilita-se a libVIP **alterada**. Estas modificações não indicam que a troca de mensagens será realizada de modo inseguro, mas simplesmente indicam alterações para melhorar o desempenho na comunicação utilizando a biblioteca VIP. Caso o usuário especificar a modalidade de comunicação confiável, são adotadas as características padrões de inicialização de um *socket*, caracterizando a libVIP **normal**.

Em implementações tradicionais das funções de *sockets*, o protocolo reúne várias pequenas mensagens em uma região de memória intermediária no núcleo do sistema operacional e quando esta alcança um determinado tamanho, faz-se a transmissão dos dados [14]. Contudo, certas aplicações desejam transmitir seus dados imediatamente após a chamada de enviar. Para isso, a primeira alteração realizada consiste em habilitar a opção `TCP_NODELAY` da conexão. A outra modificação implementada diz respeito ao nível de importância para cada pacote de troca de mensagem formado. A opção `SO_PRIORITY`, quando habilidada, define que todos os dados transmitidos possuem alta prioridade. O pacote com esta propriedade não entra no final da fila de transmissão de dados do sistema operacional, e sim no começo desta. Contudo, não são todos os equipamentos de rede que obedecem a estas opções.

Outras questões para aumentar a velocidade de transmissão foram testadas, mas não foram incluídas na implementação porque não alcançaram bom desempenho ou simplesmente porque suas propriedades não estão de acordo com as características de projeto da biblioteca. Por exemplo, existe a possibilidade de tornar o *socket* TCP assíncrono, ou seja, a função de envio ou recepção de dados não esperariam pela sincronização [17] para efetuar a troca de mensagem, o que não satisfaz a questão de segurança proposta pela biblioteca. Outras opções testadas incluem alterações no tamanho máximo da região de memória para enviar e receber dados, e nos valores dos tempos de retransmissão de dados. Contudo, a configuração padrão do *socket* TCP obteve melhor desempenho.

4. Avaliação de Resultados

Para avaliar a libVIP, foram construídas duas aplicações. A primeira, consiste em analisar o desempenho da libVIP,

através da mensuração da largura de banda e do tempo de comunicação providos pela biblioteca. A segunda aplicação realiza a transmissão de dados do fractal de Mandelbrot, com o intuito de avaliar a corretude das operações de troca de mensagens. Em ambas as aplicações, foram utilizadas as duas modalidades de comunicação da libVIP (ver subseção 3.4), a libVIP normal e a alterada.

A primeira aplicação foi executada no aglomerado de computadores do Laboratório de Sistemas de Computação (LSC) da Universidade Federal de Santa Maria. Cada máquina desse aglomerado contêm dois processadores Pentium III de 1 GHz, com 768 MBytes de memória principal. Foi utilizada a placa de rede Gigabit Ethernet modelo 3Com 3C996-T e a comunicação entre as máquinas ocorre através de um comutador Gigabit Ethernet modelo 3Com 3C17700. A aplicação envolvendo a transmissão do fractal de Mandelbrot foi testada no aglomerado do Laboratório de Tecnologia em Clusters (LabTeC) da Universidade Federal do Rio Grande do Sul. Nesse aglomerado, cada máquina possui 2 processadores Pentium III 1GHz, com 1 GByte de memória principal. As máquinas do aglomerado do LabTeC possuem adaptadores de rede Gigabit Ethernet modelo Intel 82544EI e a troca de mensagens entre elas ocorre através de um comutador Fast Ethernet 3Com 3C16986A.

4.1. Largura de Banda e Tempo de Comunicação

Esta aplicação, chamada de Ping-Pong, tem a função de medir o desempenho da biblioteca VIP nas operações de troca de mensagem. Para isso, além das duas versões desta aplicação com a libVIP, também foi construída uma versão utilizando somente *sockets* TCP. A aplicação executa em duas máquinas, uma cliente e outra servidora. O cliente envia uma mensagem para o servidor e este, ao receber a mensagem, realiza a operação de mandar de volta a mensagem para o cliente. Para cada tamanho de mensagem testado, são realizadas 100 iterações deste procedimento. O tempo da comunicação t_c e a largura de banda b medidos para um certo tamanho de mensagem m são dados pelas equações:

$$t_c = \frac{t_f \quad t_i}{2p} \tag{1}$$

$$b = \frac{m}{t_c} \tag{2}$$

onde t_i é a hora local medida antes das iterações, t_f é a hora local medida após as iterações e p é o número de iterações. A equação 1 contém um fator 2 no denominador porque a mensagem de tamanho m vai e volta do cliente ao servidor.

Os procedimentos para a troca de mensagens nas versões usando *sockets* TCP e usando libVIP para as máquinas cliente e o servidor estão apresentados respectivamente nas figuras 3 e 4. A versão libVIP foi organizada para usufruir da sua propriedade de assincronismo. Nesta versão, primeiramente são realizadas as requisições de enfileiramento de descritores, através das diretivas `VipPostSend` e `VipPostRecv`, seguidos da espera dos resultados. Pelo fato que a libVIP possui dois fluxos de execução para a troca de mensagem, um para o envio e outro para recepção, as duas requisições nesta versão podem ser efetuadas em paralelo. A versão *socket TCP* utiliza as diretivas síncronas `send()` e `recv()`.

cliente:	servidor:
repete p vezes:	repete p vezes:
send()	recv()
recv()	send()

Figura 3. Ping-Pong sockets

cliente:	servidor:
repete p vezes:	repete p vezes:
VipPostSend()	VipPostRecv()
VipPostRecv()	VipPostSend()
VipSendWait()	VipRecvWait()
VipRecvWait()	VipSendWait()

Figura 4. Ping-Pong libVIP

Foram realizados testes com mensagens variando de 10 a 100.000 Bytes. Os resultados estão apresentados em duas figuras, com dois gráficos cada. Os gráficos da figura 5 apresentam uma visão de toda a análise efetuada, enquanto que os gráficos da figura 6 mostram um detalhe deste gráfico para o trecho das mensagens pequenas, de até 10.000 Bytes.

Na figura 5 observa-se que a libVIP normal consegue melhor largura de banda, medida em 41,1 MBytes/s. Esta versão atinge melhores resultados com o tamanho da mensagem variando entre 20 e 40 KBytes. A libVIP alterada apresenta o valor de 38,75 MBytes/s e também alcança melhores resultados no intervalo de valores citado para a versão da biblioteca VIP normal. A versão TCP produz maior largura de banda quando transmite mensagens de 40 KBytes de tamamho, com a taxa de 35 MBytes/s. O melhor desempenho obtido pela libVIP deve-se ao fato que as operações de troca de mensagem com essa biblioteca são executadas em paralelo, ao contrário da aplicação envolvendo somente o *sockets* TCP, que realiza as chamadas `send()` e `recv()` sincronamente. Após a transmissão de dados acima de 40 KBytes, as versões utilizadas para a avaliação tendem a estabilização, provendo aproximadamente 33.5 MBytes/s de vazão. Apesar dos índices de largura de banda apresentarem variações significativas entre as versões, o tempo de comunicação evolui praticamente da mesma maneira entre as versões de protocolos utilizados para a análise.

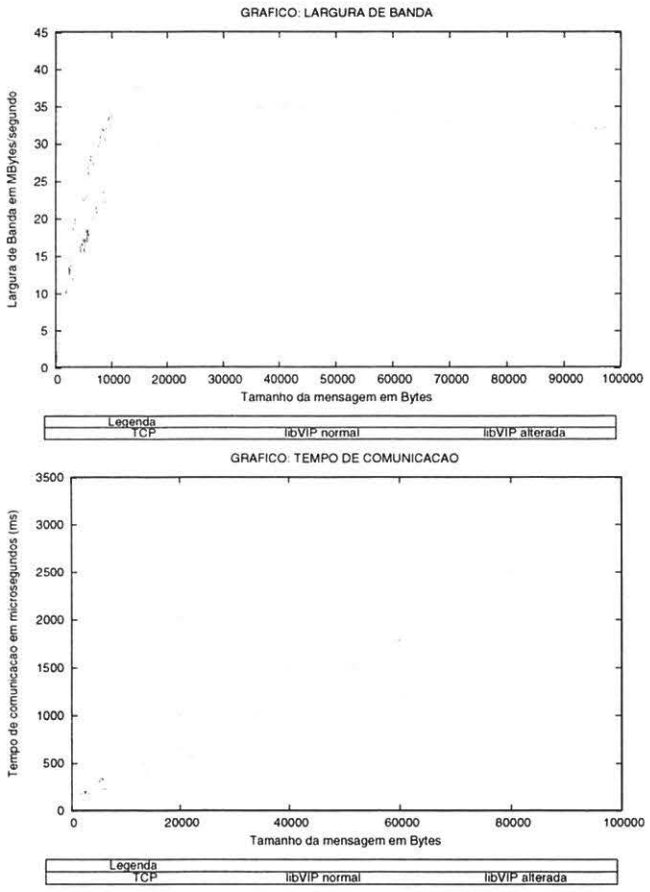

Figura 5. Mensagens de até 100.000 Bytes

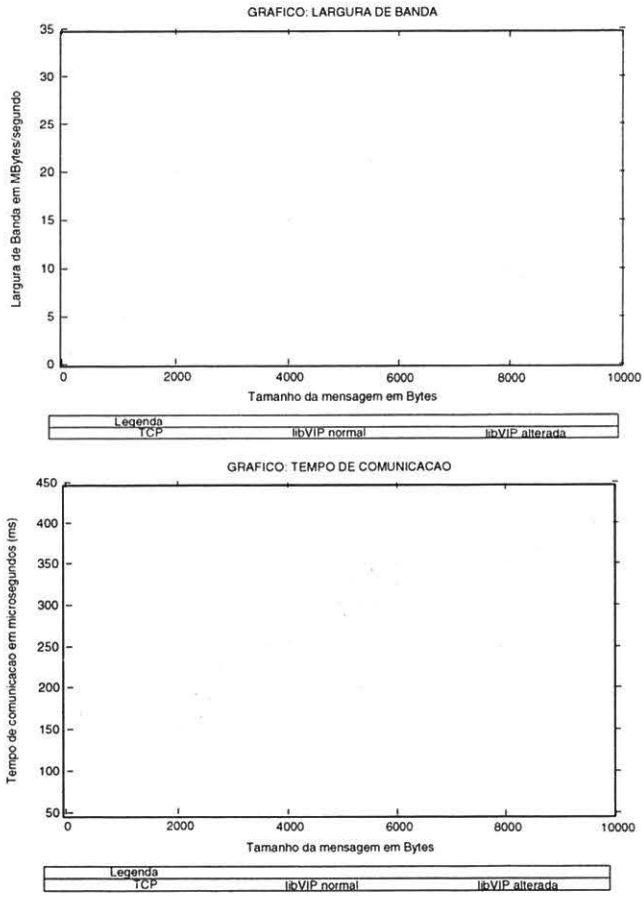

Figura 6. Mensagens de até 10.000 Bytes

A figura 6 mostra que as duas modalidades de comunicação da biblioteca VIP atingem aproximadamente 33 MBytes/s, ao transmitir mensagens de 10 KBytes. A versão TCP apresenta melhores resultados que a libVIP normal até a passagem de mensagem de 2400 Bytes; contudo, após este valor a versão TCP sofre um decréscimo da largura de banda. O melhor índice desta versão é de 24.6 MBytes/s, quando opera com mensagens de 10 KBytes.

4.2. Geração de Fractal

Esta aplicação realiza a interação entre vários processos para o cálculo do fractal de Mandelbrot. São calculados os valores de cores para uma janela gráfica de 800x800 pontos. O desenho do fractal de Mandelbrot é sempre determinístico, o que possibilita inferir sobre a corretude dos dados no processo de troca de mensagens.

Esta aplicação emprega o conceito de mestre-escravo [15]. O mestre passa para os escravos o número de uma linha da janela gráfica a ser calculada por ele. O escravo realiza o cálculo das cores dos pontos da linha, envia os resultados para o mestre e espera por mais uma requisição de trabalho.

A aplicação de Mandelbrot, foi construída sob dois diferentes modelos mestre-escravo, apresentados na figura 7. O modelo da figura 7a, denominado modelo A, efetua a comunicação entre duas interfaces virtuais explorando a técnica de fluxos de execução na máquina do escravo. Cada fluxo de execução no escravo solicita trabalho, calcula os valores de cores e devolve os resultados para o mestre. Para controlar que só um fluxo de execução faça uso da interface virtual em um dado instante, é empregado o mecanismo de exclusão mútua para a comunicação.

Já na figura 7b têm-se o modelo B, onde o mestre cria n interfaces virtuais que são conectadas a n escravos. Cada escravo pode trocar mensagens com o mestre através da sua própria interface virtual e não são necessários métodos de exclusão mútua para a comunicação, visto que cada escravo possui somente um fluxo de execução.

Para a realização dos testes, foram adotados as duas modalidades de comunicação da libVIP e os dois modelos mestre-escravo apresentados na figura 7. Para as aplicações envolvendo o modelo B, cada par de escravos é executado em uma máquina do aglomerado, visto que esta dispõe de dois processadores. Com a execução dos testes, observou-se que

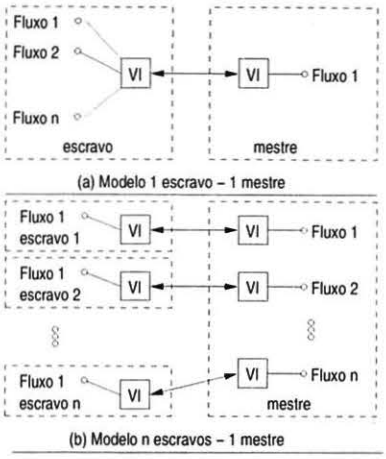

Figura 7. Modelos Mestre-Escravo para a Aplicação de Mandelbrot

as duas modalidades de comunicação da libVIP, em ambos os modelos mestre-escravo, transmitiram os dados do fractal de Mandelbrot com total precisão.

Tabela 3. Modelo A, um escravo e um mestre

Número de Fluxos	Tempo (s) com libVIP normal	Tempo (s) com libVIP alterada
1	14.96	14.96
2	7.43	7.43
4	7.41	7.41
8	7.40	7.40
16	7.39	7.39
32	7.41	7.41

Tabela 4. Modelo B, vários escravos e um mestre

Número de Escravos	Tempo (s) com libVIP normal	Tempo (s) com libVIP alterada
1	15.25	14.92
2	7.45	7.45
4	3.87	3.87
8	2.02	2.02
16	1.34	1.33
32	1.05	1.04

As tabelas 3 e 4 apresentam, respectivamente, os índices de tempo obtidos na execução dos modelos A e B. Na aplicação para o modelo A, pode-se notar que com a utilização de 2 fluxos de execução, têm-se um fator de aceleração bastante significativo, pois a máquina escravo é biprocessada. Contudo, com a adição de mais fluxos, a eficiência diminui e o acrésimo de desempenho é mínimo. Isto deve-se ao fato que a interface virtual no escravo opera como uma região crítica e, quanto maior é o número de fluxos, maior é a sobrecarga no tratamento da variável de exclusão mútua que trata a interface virtual. Nesse modelo, o melhor índice de desempenho é alcançado com 16 fluxos de execução.

A tabela 4 apresenta os resultados para o modelo B. Observando-se os resultados obtidos, nota-se que com a utlização de n escravos, onde n varia de 2 a 16, obtêm-se aproximadamente uma aceleração de ordem n. O desempenho atingido deve-se ao fato que um escravo (um fluxo de execução) opera sobre um processador e possui uma conexão direta com o mestre.

5. Conclusão

A arquitetura de aglomerados de computadores tem emergido como um ambiente de execução viável e de baixo custo para aplicações que demandam por alto poder de processamento. Para melhorar o desempenho de aplicações nesse tipo de máquina paralela, tem-se buscado métodos para otimizar os protocolos de comunicação entre computadores. Baseado nesse contexto, foi lançado em 1997, a especificação da Arquitetura de Interface Virtual (VIA). VIA define um protocolo de alto desempenho para ser utilizado em redes de sistema e sua principal propriedade consiste em realizar a transmissão de dados sem a interferência do sistema operacional. Para VIA funcionar, é necessário para cada adaptador de rede, um programa VIA dentro do núcleo do sistema operacional para interagir diretamente com a interface de rede.

Baseado na arquitetura VIA, foi projetada e implementada a biblioteca VIP, ou libVIP. Esta biblioteca foi construída para sistema operacional Linux e implementa um subconjunto de 12 procedimentos disponibilizados na especificação VIA. As principais características da libVIP são a segurança nas operações de troca de mensagem e a portabilidade. A biblioteca funciona em quaisquer computadores que possuam adaptadores de rede configurados para trabalhar com o protocolo TCP/IP, dispensando a utilização de um módulo VIA para cada adaptador.

A libVIP está baseada no modo de comunicação assíncrono. Esta é uma propriedade interessante para esconder o tempo de latência de comunicação entre computadores. Para ganhar em desempenho, pode-se utilizar a biblioteca VIP da seguinte forma: envia-se os dados, realiza-se alguma computação útil e depois espera-se pela confirmação de envio dos dados. Outro modelo relevante é realizar primeiramente todas as postagens de descritores para depois realizar todas as esperas pelos resultados.

Os testes com a aplicação Ping-Pong demonstram que a libVIP atinge melhor largura de banda que a versão *sockets*

TCP, pois explora a sua propriedade de possuir dois fluxos de execução, que operam em paralelo com as diretivas da biblioteca, para realizar a troca de mensagens. Isto possibilita a libVIP mandar e receber mensagens ao mesmo tempo. A libVIP alterada atinge melhores resultados com mensagens de até 10 KBytes e, a libVIP normal é mais eficiente para mensagem acima deste valor. Com a execução da aplicação do fractal de Mandelbrot, observa-se que a biblioteca VIP transferiu os dados entre duas interfaces virtuais com total precisão. Além disso, esta aplicação também apresenta bons índices de tempo com o modelo B, vários escravos e um mestre, onde o desempenho é diretamente proporcional ao número de máquinas utilizadas.

Como trabalhos futuros para a libVIP, existe a possibilidade de realizar o porte desta biblioteca para o ambiente Windows. Além disso, pode-se implementar uma versão da biblioteca sobre o protocolo de transporte UDP (*User Datagram Protocol*), que é mais veloz que o TCP, desenvolvendo uma camada de *software* para garantir a confiabilidade. Também pode-se citar como um trabalho futuro, o estudo da especificação da arquitetura Infiniband [1], que desponta como uma sucessora da Arquitetura de Interface Virtual. O Laboratório LabTeC possui quatro adaptadores de rede e um comutador que empregam a tecnologia Infiniband. Existe planos de implementar uma bibliboteca de comunicação usando diretamente a API proporcianada na especificação Infiniband, visto que o *hardware* se encontra disponível.

Referências

[1] BARCELLOS, M. P.; GASPARY; L. P. *Tecnologias de Rede para Processamento de Alto Desempenho* ERAD 2003 - Escola Regional de Alto Desempenho, Vol 1, pp. 58-102, jan. 2003.

[2] BENNETT, J. K.; ABDEL-SHAFI, H.; SPEIGHT, E. *Realizing the Performance Potential of the Virtual Interface Architecture*. In Proceedings of International Conference on Supercomputing, Jun., 1999. `http://citeseer.nj.nec.com/speight99realizing.html`

[3] BUYYA, R. *High Performance Cluster Computing*, Ed. Prentice Hall PTR, Vol 1. *Architecture and Systems*, 1999.

[4] COMMER, D. E. *Redes de Computadores. Transmissão de dados, ligação inter-redes e Web*, Ed. Bookman, Vol. 2, Porto Alegre/RS, 2001.

[5] Compaq Computer Corp.; Intel Corp.; Microsoft Corp. *Virtual Interface Architecture Specification*. Version 1.0, Dec. 1997. `www.cs.cornell.edu/barr/repository/cs614/san\_10.pdf`.

[6] DRUSCHEL, P. *Operating System Support for High-Speed Networking*. In Communications of the ACM, 39(9), pp. 41–51, Sep. 1996.

[7] EICKEN, E. V.; VOGELS, W. *Evolution of Virtual Interface Architecture*. IEEE Computer Magazine, pp. 61-68, Nov. 1998.

[8] Intel Corporation. *Virtual Interface Architecture. Defining de Path High-Performance Scalable Clusters*, 1997. `www-old.cs.umn.edu/computing-resources/dell-cluster/report.pdf`.

[9] KERNIGHAN, B. W.; RITCHIE, D. M. *C a Linguagem de Programação Padrão ANSI*. Vol. 7, Ed. Campus, Rio de Janeiro, 1989.

[10] KLEIMAN, S.; SHAH, D.; SMAALDERS, B. *Programming with Threads*. Ed. Prentice Hall, 1996.

[11] LOBOSCO, M.; SILVA, A.; SANTOS, V.; AMORIM, C. *TCP/IP versus VIA on Network of Workstation*, SBAC-PAD. 13th Symmposium on Computer Architecture and high Performance Computing, Vol. 13, pp. 140-147, 2001.

[12] BUONADONNA, P.; GEWEKE, A.; CULLER, D. *An implementation and Analisys of the Virtual Interface Machine*, In Proceedings of SC '98, Orlando, Florida, pp. 7-13, Nov. 1998.

[13] Information Technology - Portable Operating System Interface (POSIX) 1003.1-1990 Part 1: System Application Program Interface (API) C Language, 1990. `http://standards.ieee.org/cgi-bin/status`.

[14] TANENBAUM, A. S. Redes de Computadores, Ed. Campus, Vol. 3, Rio de Janeiro/RJ, 1997.

[15] TANENBAUM, A. S. Sistemas Operacionais, Ed. Bookman, Vol. 3, Rio de Janeiro/RJ, 1999.

[16] RIGHI, R. R. *libVIP - Desenvolvimento em Nível de Usuário de uma Biblioteca de Comunicação que Implementa o Protocolo de Interface Virtual*, Trabalho de Graduação nº 163. Curso de Ciência da Computação, UFSM. Santa Maria/RS, Fev. 2003.

[17] WILKINSON, B.; ALLEN, M. *Parallel Programming: techniques and applications using networked workstations and parallel computer*Ed. Pretice Hall, 1999.

[18] GAY, W. W.*Linux Sockets Programming by Example*.Ed. QUE, 2000.