Universidade Federal do Paraná

Departamento de Informática

Leslie H Watter, Marcelo L Stival, Roberto A Hexsel

Avaliação de Desempenho do Protocolo IEEE 802.2-LLC no Kernel do Linux

Relatório Técnico
RT-DINF 002/2007

Curitiba, PR
2007

Avaliação de Desempenho do Protocolo IEEE 802.2-LLC no Kernel do Linux

Leslie H Watter, Marcelo L Stival, Roberto A Hexsel

Departamento de Informática
Universidade Federal do Paraná
Centro Politécnico, C Postal 19081 – 81531-990 Curitiba, PR
roberto@inf.ufpr.br

Abstract: *This report describes a performance analysis of the protocol IEEE 802.2-LLC implemented on a separated tree of Linux kernel. The analysis was done in a cluster of computers using the* OpenMPI *library. A new especific component was developed to support the IEEE 802.2-LLC protocol by the* OpenMPI *library, in a way that the applications do not need to be modified to use the new protocol. The results of the performance analysis showed that the utilization of IEEE 802.2-LLC protocol result in a reduction of time of the message passing subsystem in a comparison with the TCP/IP protocols.*

Resumo: *Este relatório descreve uma avaliação de desempenho do protocolo IEEE 802.2-LLC implementado em uma árvore de testes no kernel do Linux. A avaliação foi desenvolvida em um ambiente composto por um aglomerado de computadores utilizando a biblioteca* OpenMPI*. Foi desenvolvido um componente específico para o suporte ao protocolo IEEE 802.2-LLC na biblioteca* OpenMPI*, de modo que as aplicações paralelas não precisassem ser modificadas para fazer uso deste novo protocolo. Os resultados da avaliação de desempenho mostraram que a utilização do protocolo IEEE 802.2-LLC trouxe redução no tempo de troca de mensagens, comparado com a utilização dos protocolos TCP/IP.*

1 Introdução

A demanda por poder computacional em aplicações científicas tem sido crescentemente suprida por aglomerados de PCs, interligados por uma rede local. Bibliotecas de troca de mensagens como MPI [9] e PVM [19] são usadas para a programação das aplicações segundo o modelo de Troca de Mensagens em Multicomputadores [6].

As bibliotecas de comunicação são implementadas para executarem sobre os protocolos TCP/IP [17]. O protocolo TCP [18] provê comunicação confiável entre computadores localizados em quaisquer locais servidos por uma rede com protocolo similar ao IP. Por conta da incerteza na localização dos computadores, o TCP incorpora uma série de mecanismos para garantir a entrega confiável das mensagens, bem como para manter a rede operando em condições razoáveis de tráfego e sem congestionamento.

Se um aglomerado é instalado num único prédio ou numa sala, os computadores possivelmente estarão interligados por uma rede local, que na grande maioria dos casos é uma

rede Ethernet (ou IEEE 802.3*). O protocolo de enlace da Ethernet é o padrão IEEE 802.2 (*Logical Link Control – LLC*), e este incorpora uma série de funcionalidades que são similares ou equivalentes a um sub-conjunto das funcionalidades do TCP. O escopo geográfico do LLC é, por projeto, menor que o do TCP, e o meio no qual o LLC transmite seus dados (sub-camada MAC) é implementado em hardware nos adaptadores de interface de rede. Em PCs de baixo custo, a implementação do TCP (e do IP) é inteiramente em software. Por conta destas duas características, é de se supor que um aglomerado de PCs construído sobre uma biblioteca que utilize LLC como seu protocolo de transporte seja mais eficiente do que um aglomerado baseado na mesma biblioteca mas com TCP como seu protocolo de transporte. Isso decorre das diferentes implementações: os protocolos TCP/IP são implementados com funcionalidades para garantir entrega confiável através de redes diversificadas, enquanto que o LLC é restrito a uma rede local e o controle da integridade dos dados é responsabilidade do adaptador de rede, que tem o algoritmo de detecção de erros (CRC) implementado em hardware.

Independente da arquitetura empregada na construção de um aglomerado de computadores, e da técnica utilizada na aplicação, todas tem um ponto de convergência: a necessidade de troca de mensagens entre as diferentes instâncias que estão executando a aplicação. Estas instâncias podem ser computadores completos ou apenas processadores com memória compartilhada, aonde todos tem que, em algum momento, se comunicar para trocar resultados e/ou sincronizar o trabalho em algum ponto.

Tamanha é a necessidade de troca de informações que ela pode facilmente tornar-se o gargalo na computação do resultado. Têm-se procurado incessantemente reduzir a carga adicionada à computação pelo número de troca de mensagens, ora procurando minimizar as trocas, ora buscando reduzir o tempo agregado à passagem da mensagem de um ponto a outro do sistema.

Dessa maneira, os estudos voltaram-se para a tentativa de eliminar e/ou reduzir ao máximo o tempo gasto na troca de mensagens. Alguns apontam para os protocolos intermediários como sendo os grandes vilões na utilização do tempo necessário para a troca de mensagens [24, 14, 3], enquanto outros apontam para o hardware, e partem em busca de melhorá-lo de maneira a deixar o processador mais tempo livre para executar o processamento [13].

Para reduzir a sobrecarga imposta pelo software, é necessário verificar os protocolos que transportam as mensagens. Aqueles que apontam para os protocolos como sendo os vilões no consumo do tempo de processamento na troca de mensagens normalmente não o fazem apontando para o protocolo como um todo, mas sim para a cópia de dados necessária no processamento desses protocolos [21, 16].

Objetivando reduzir a sobrecarga imposta pelos protocolos de comunicação, mais especificamente os protocolos TCP/IP em um aglomerado de computadores, o presente estudo buscou um protocolo de comunicação alternativo que fornecesse as funcionalidades necessárias e com menor sobrecarga. A alternativa encontrada foi utilizar o protocolo IEEE 802.2-LLC, implementado à parte, em uma árvore de desenvolvimento do kernel do Linux separada da árvore principal.

O trabalho desenvolvido em [7] serviu de base para o presente estudo. Neste trabalho [7], as modificações efetuadas no kernel do Linux, mais especificamente no protocolo LLC permitem que uma interface de soquete que utilize o protocolo LLC seja criada no espaço de usuário, o que antes não era possível.

Para caracterizar as diferenças incorridas na mudança de protocolo de comunicação, dentro de um aglomerado de computadores, se fez necessário utilizar uma biblioteca de comunicação paralela. Para o efetuar o estudo descrito aqui, foi necessário modificar a biblioteca paralela OpenMPI, incluindo um novo componente, responsável pela comunicação utilizando o protocolo LLC. Após o trabalho de desenvolvimento na biblioteca OpenMPI, foi efetuada a análise de desempenho dos aplicativos NetPIPE, MPPTEST e FFT variando-se apenas o protocolo de comunicação subjacente (LLC e TCP/IP). Dessa avaliação obtemos resultados que apontam ganhos de desempenho do protocolo LLC com relação ao TCP/IP da ordem de 3% para mensagens grandes e 15% para mensagens pequenas.

2 Troca de Mensagens

A utilização do protocolo LLC pode ser entendida como um agente facilitador na comunicação dentro de aglomerados, independentemente da arquitetura de rede utilizada. O objetivo dos modelos existentes tem sido o de melhorar o desempenho especificamente reduzindo a latência, a sobrecarga na comunicação, além de aumentar a largura de banda [5]. A figura 1 ilustra o caminho que a mensagem percorre quando é transmitida de um ponto a outro da rede.

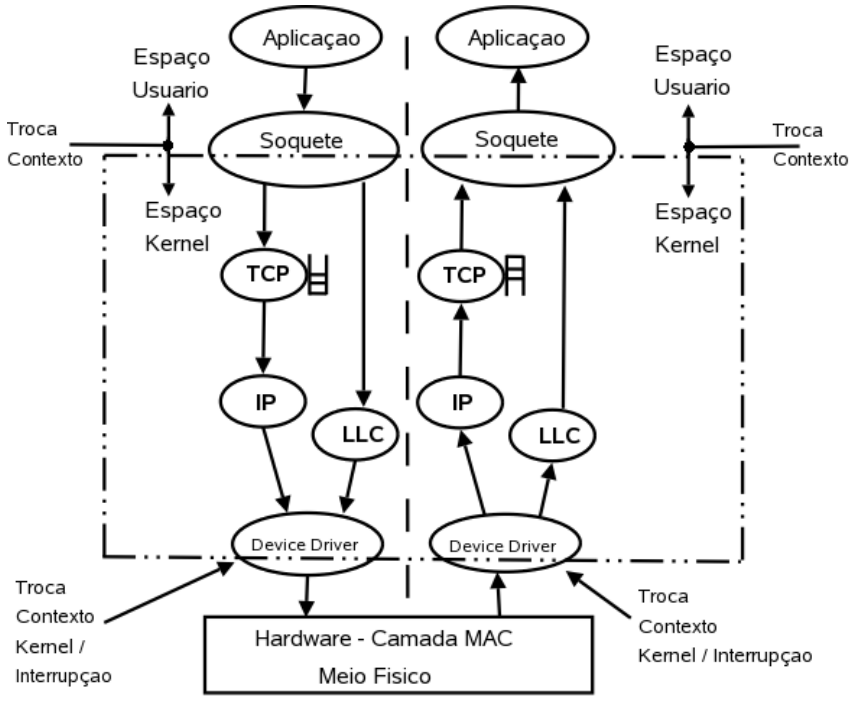

Figure 1: Caminho de uma mensagem através do Kernel

Diferentemente do protocolo TCP, o protocolo LLC não identifica seus processos pelo uso de portas lógicas, ao invés disso, são identificados através das portas de acesso a serviço (*Service Access Point* – SAP). Uma porta de acesso ao serviço é um endereço de 8 bits que pode indicar um único processo ou um grupo de processos.

Uma porta de acesso a serviço é equivalente a uma porta para o protocolo da camada de transporte. Se a rede usa múltiplos protocolos, cada protocolo da camada de rede terá a

sua própria SAP. Esse é o método utilizado pelo LLC para identificar qual protocolo está conversando com qual outro.

2.1 Comparação dos protocolos LLC e TCP/IP

A seguir é estabelecida uma correlação entre as características do TCP/IP e do LLC/MAC, de maneira a esclarecer como é possível, dentro dos limites de uma rede local, substituir os protocolos TCP/IP pelo protocolo LLC/MAC. A tabela 2 mostra uma comparação entre o modelo estratificado dos protocolos empregados na Internet e a implementação de ambos os protocolos (LLC e TCP/IP) no kernel do Linux.

<table>
<tr><th colspan="2">Modelo TCP/IP Estratificado</th><th colspan="2">Implementação no Kernel do Linux</th></tr>
<tr><th>Camada</th><th>Protocolo</th><th>TCP/IP</th><th>LLC</th></tr>
<tr><td colspan="2">Aplicação</td><td colspan="2">Aplicação</td></tr>
<tr><td>Transporte</td><td>TCP</td><td>Soquete TCP</td><td></td></tr>
<tr><td>Rede</td><td>IP</td><td>Soquete IP</td><td></td></tr>
<tr><td>Enlace</td><td>LLC</td><td></td><td>Soquete LLC</td></tr>
<tr><td rowspan="2">MAC</td><td rowspan="2"></td><td colspan="2">Device Driver</td></tr>
<tr><td colspan="2" rowspan="2">Dispositivo de Rede</td></tr>
<tr><td>Física</td><td></td></tr>
</table>

Figure 2: Comparação Entre o Modelo Estratificado e a Implementação no Kernel

No tocante ao endereçamento de serviços através da interface de soquetes, o TCP/IP emprega pares <endIP, porta> para endereçar as duas pontas de um enlace lógico, enquanto que LLC/MAC emprega pares <endMAC, SAP>. Para cada *endereço IP* de uma estação, utiliza-se o *endereço MAC* da interface de rede associada àquela estação. Para cada *porta* de serviço TCP, é utilizada uma *porta de acesso a serviço* do protocolo LLC, mantidas as devidas restrições apontadas na tabela 1).

Funcionalidades	**TCP/IP**	**LLC**
Endereçamento de pacotes	inter-redes	rede local
Garantia de integridade	checksum dos cabeçalhos TCP/IP	CRC do quadro Ethernet
Largura das janelas deslizantes	2^{32}	2^{8}
Número de portas	65.536	128
Tam. máx. do pacote de dados	65.535 bytes	depende do MTU

Table 1: Comparação Funcionalidades TCP/IP e LLC

Embora o protocolo TCP/IP possa ser usado em praticamente qualquer tipo de rede, este estudo limita-se à rede local, uma vez que a estrutura predominante nos aglomerados de computadores é a rede local. A garantia de transferência íntegra dos dados de ambos os protocolos de uma estação à outra decorre do cálculo do CRC do quadro Ethernet.

Assim, o cálculo do *checksum* dos cabeçalhos TCP/IP é redundante (em se tratando de redes locais) porque a taxa de erros em redes Ethernet é extremamente baixa.

Para aumentar a vazão obtida na transmissão de dados, ambos os protocolos implementam a técnica de janelas deslizantes, embora os tamanhos de janela sejam muito diferentes.

Enquanto que no TCP/IP uma *porta* se refere a uma *porta lógica*, no LLC essa definição é utilizada para *porta de acesso a serviço*. Independente da nomenclatura utilizada, ambas as portas são utilizadas para multiplexação de serviços nas camadas superiores.

No TCP/IP é possível enviar um pacote de tamanho maior que o MTU da rede, porque a fragmentação dos pacotes é feita transparentemente pelo protocolo IP. Usando o LLC, o tamanho máximo de um pacote depende do MTU da rede utilizada, descontados os tamanhos do cabeçalho do protocolo físico e do LLC. Na implementação do LLC no kernel do Linux, o próprio protocolo LLC implementa a fragmentação e controle de envio dos dados.

3 Bibliotecas Paralelas

A troca de mensagens é o método de comunicação baseado no envio e recebimento de mensagens através de uma rede de computadores seguindo as regras de um protocolo de comunicação.

Os padrões de troca de mensagens MPI [9] e PVM [19, 23], são os padrões mais utilizados na atualidade como *frameworks* para a criação e execução de aplicações paralelas. Para este estudo, foi escolhido utilizar uma biblioteca de troca de mensagens (MPI) que fornecesse uma abstração da realidade de hardware para que o software interagisse com todo o aglomerado de computadores.

Devido à limitação da implementação atual do protocolo LLC não fornecer a interface de *loopback*, a escolha da biblioteca de troca de mensagens obedeceu à essa restrição. A implementação OpenMPI 1.0 [11, 10] foi o padrão MPI escolhido para ser portado para utilização do protocolo LLC nesse estudo. Essa implementação foi escolhida por diferentes fatores, como por exemplo, modularidade, flexibilidade, versão em constante desenvolvimento e implementação que suporta os padrões MPI versões 1 e 2 e principalmente, adequar-se à restrição da implementação do protocolo LLC.

Para uma melhor compreensão do trabalho desenvolvido é necessário entender como está estruturada a biblioteca OpenMPI. Existem três tipos de *frameworks* na biblioteca Open-MPI [11, 10, 17], aqueles que estão na camada MPI (OMPI), os que se encontram na camada de execução (ORTE) e ainda os que estão presentes na camada que fica em contato com o sistema operacional e a plataforma de destino (OPAL).

O *framework* BTL é o responsável por criar a camada de abstração de transferência de dados entre estações ponto-a-ponto. Todas as implementações do *framework* BTL que tratam dos protocolos subjacentes, como por exemplo TCP/IP e LLC, são chamadas de componentes BTL. Foi desenvolvido um componente BTL específico para o protocolo LLC.

Devido ao fato de o protocolo LLC utilizado (nível 2) operar de maneira semelhante ao protocolo TCP, a estrutura base da implementação do componente BTL LLC seguiu a estrutura existente do componente BTL TCP, observadas as necessárias modificações a respeito de endereçamento e características próprias de cada protocolo. Desta maneira,

utilizando uma estrutura comum, fica fácil estabelecer uma comparação entre os protocolos apresentados utilizando aplicações que empreguem a biblioteca OpenMPI.

Para avaliar a implementação do componente BTL LLC e a implementação do protocolo LLC no *kernel*, optou-se por executar aplicações que empregam a biblioteca OpenMPI. Essas aplicações compreendem dois grupos distintos: dois *benchmarks* de rede (NetPIPE e MPPTEST) e uma aplicação computacionalmente intensiva com grande número de troca de mensagens (*implementação da transformada rápida de fourier*). Estas são brevemente descritas a seguir.

NetPIPE – *Network Protocol Independent Performance Evaluator* É uma ferramenta desenvolvida para representar visualmente o desempenho da rede sob uma grande variedade de condições [22]. Essa ferramenta executa um teste ping-pong entre duas máquinas, enviando mensagens cujos tamanhos crescem gradativamente, seja entre dois processos que estão em máquinas separadas por uma rede, seja em uma máquina multiprocessada. Cada ponto amostrado envolve vários testes ping-pong[1] executados para medir com precisão o tempo.

MPPTEST – *Measuring MPI Performance* O programa MPPTEST [12] tem a finalidade de medir o desempenho de algumas rotinas de troca de mensagens do padrão MPI em uma grande variedade de situações. Além do teste ping-pong, pode ser medido o desempenho com vários processos participando do teste (expondo assim problemas de escalabilidade e contenção) e pode-se adaptativamente escolher os tamanhos de mensagens de maneira a isolar mudanças bruscas no desempenho.

Fast Fourier Transform A implementação da transformada rápida de Fourier [1] utilizada nesse estudo é a mesma utilizada no trabalho [8]. O objetivo aqui é comparar os tempos obtidos na execução da aplicação, com os dois protocolos de comunicação TCP/IP ou LLC, de forma a prover uma base de comparação para aplicações já existentes.

4 NetPIPE

Utilizou-se o NetPIPE para comparar a eficiência dos protocolos subjacentes à biblioteca OpenMPI. Dessa maneira, utilizando a mesma biblioteca e variando apenas o protocolo, é possível avaliar o desempenho do protocolo no que se refere à execução de aplicações paralelas.

O NetPIPE executa um teste ping-pong, enviando mensagens de vários tamanhos entre dois processadores. Os tamanhos de mensagem são escolhidos a intervalos regulares de tempo e cada ponto amostrado envolve vários testes ping-pong para fornecer uma medida de tempo precisa.

A figura 3, que ilustra os resultados do NetPIPE, mostra no eixo x (em escala logarítmica) o tamanho da mensagem em *bits* e no eixo y o tempo em *microsegundos* decorrido para a execução dos testes.

[1] Em um teste ping-pong, uma mensagem é enviada da máquina A para a máquina B e, ao receber a mensagem, a máquina B devolve-a para a máquina A.

A figura 3 mostra o tempo necessário para o envio de um pacote de dados de um nó a outro da rede, executando um teste ping-pong. Nesta mesma figura, encontram-se além das duas linhas dos protocolos LLC e TCP/IP uma terceira linha cujos valores indicam a porcentagem de ganho do LLC em relação ao TCP/IP, qual é a diferença do percentual de tempo em que o LLC é mais rápido que o TCP/IP.

Observa-se que para mensagens de tamanho inferior a 100 bits o ganho do LLC chega perto de 20%, caindo em seguida à medida que o tamanho da mensagem aumenta. É possível observar que, mesmo para um tamanho de mensagem de 8192 bits (1024 bytes) o protocolo LLC ainda tem um ganho de cinco pontos percentuais.

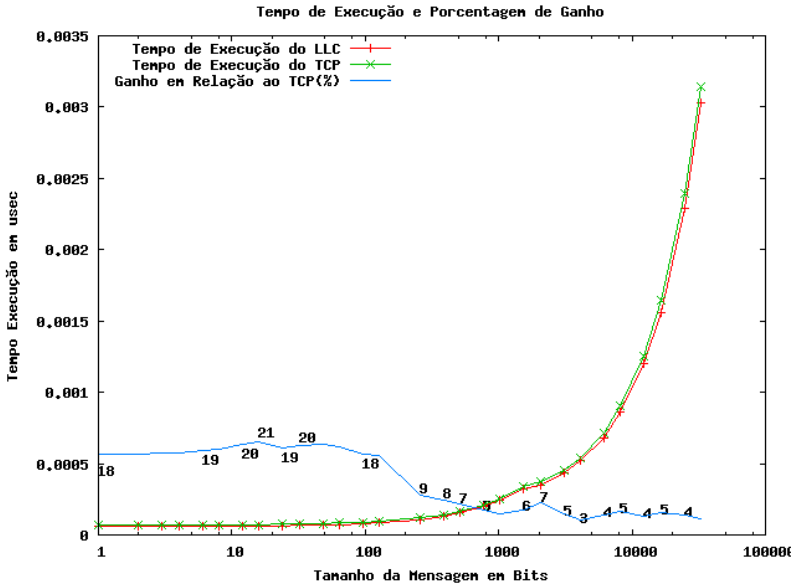

Figure 3: Tempo de Execução

A figura 4 ilustra os resultados do NetPIPE mostrando no eixo x (em escala logarítmica) o tamanho da mensagem em *bits* e no eixo y a vazão atingida em *megabits/s*.

Observa-se nas figuras 3 e 4 que o ganho do LLC para mensagens de tamanho inferior a 100 bits é alto com relação às mensagens maiores. Esse fato é explicado porque, ao contrário do TCP, o LLC não emprega um buffer de transmissão de dados. Sendo assim, é possível observar que a partir do momento em que o tamanho da mensagem cresce, preenchendo-se o buffer de saída do TCP mais rapidamente, a porcentagem de ganho do LLC se reduz até estabilizar-se em um patamar ao redor de 5%. Devido ao algoritmo utilizado pelo TCP para otimizar a transmissão de mensagens, e também o fato de o tamanho de buffer utilizado pelo TCP no Linux ser variável, observa-se que para mensagens menores aonde é necessário um maior número de mensagens/dados para preencher o buffer do TCP o ganho do protocolo LLC é maior. Entretanto, pode-se observar que mesmo com a utilização de mensagens maiores, o protocolo LLC continua sendo mais rápido.

Além do tempo de execução, convém observar a vazão dos dados na rede, uma vez que quanto maior a capacidade de transmissão de dados, maior a eficiência do protocolo. A figura 4 ilustra o comportamento da vazão (em megabits por segundo) atingida utilizando-se o NetPIPE.

Assim como na figura 3, é possível observar que para tamanhos de mensagens inferiores a

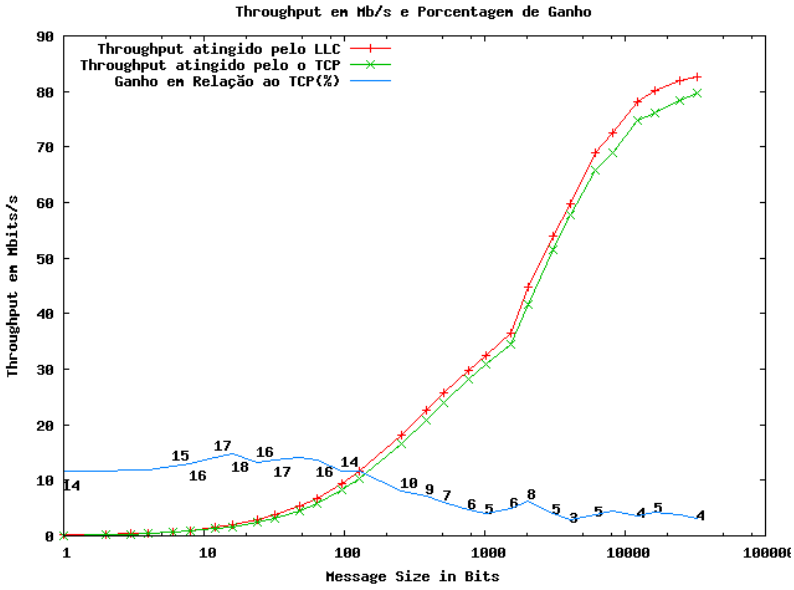

Figure 4: Vazão Atingida

100 bits o ganho do LLC é maior, de aproximadamente 15%, descrescendo à medida em que o tamanho do bloco aumenta, porém permanecendo em um patamar de 5% de ganho para os tamanhos ao redor de 8192 bits (1024 bytes).

É importante notar que os resultados produzidos pelo NetPIPE tem como fator limitante na execução do teste o tempo de envio das mensagens. O NetPIPE tenta enviar o máximo de mensagens possível dentro do intervalo de um segundo. O comportamento observado nas regiões em que os tamanhos de mensagens variam de 100 a 1000 bits e de 5000 a 10000 bits, onde o ganho do LLC reduz-se, é explicado pelo fato de o NetPIPE preencher o *buffer* de envio dos protocolos TCP/IP.

Em resumo, destes testes conclui-se que para tamanhos pequenos de mensagens, de até 100 bits, o *buffer* de envio do TCP influencia consideravelmente no tempo de transmissão dos dados, retardando o envio e portanto dando vantagem ao LLC, que não se utiliza de *buffer* de envio de dados. Para mensagens maiores que 100 bits, os ganhos sào da ordem de 3 a 8%.

5 MPPTEST – Measuring MPI Performance

O programa (MPPTEST [12]) tem a finalidade de medir o desempenho de algumas rotinas de troca de mensagens do padrão MPI. Embora distribuído juntamente com a biblioteca MPICH (uma implementação do padrão MPI), o MPPTEST segue algumas regras previamente estabelecidas com o objetivo de poder comparar as diferentes implementações do padrão MPI; dessa maneira foi possível utilizá-lo para avaliar a implementação OpenMPI.

A regra fundamental da execução dos testes com o MPPTEST é que o teste possa ser repetível, e portanto, a execução do mesmo teste várias vezes deve retornar, com um erro experimental, o mesmo resultado. Um fato bem conhecido é que a execução do mesmo programa pode produzir resultados bastante diferentes a cada vez que ele é executado. “O

único tempo que é reprodutível é o tempo mínimo de um número de testes", e essa foi a métrica escolhida para ser medida nos testes do MPPTEST [12].

Nos gráficos apresentados nesta seção (5, 6, 7 e 8), o eixo x mostra a variação no tamanho da mensagem (em *bytes*) e o eixo y mostra a variação no tempo de transmissão da mensagem de um nodo a outro (em *microsegundos*).

Observando os gráficos 5, 6, 7 e 8, que representam o mesmo teste para 2,4,8 e 16 processadores, distingüe-se três regiões com comportamentos distintos:

$[0, 1500]$ **bytes** onde se observa um crescimento quase linear na vazão medida pelo MPPTEST. Esse crescimento pode ser explicado pelo fato de o MTU da *Ethernet* ser de 1500 bytes. Assim, todas as mensagens com tamanho[2] até 1500 bytes cabem em um quadro único, evitando a fragmentação das mensagens em vários quadros.

$[1500, 1800]$ **bytes** esse intervalo mostra uma certa "instabilidade", pelo fato de se estar utilizando tamanho de mensagens com limite inferior muito próximo do *MTU*. Dessa maneira, a mensagem não cabe em um único quadro de dados, sofrendo fragmentação. A quantidade de dados restante para o segundo quadro é pequena em comparação com o tamanho total do quadro, o que eleva o custo da fragmentação para esse segundo quadro, aumentando assim o tempo de processamento relativo ao envio da mensagem.

$[1800, 4200]$ **bytes** nesse terceiro intervalo observa-se que volta a ocorrer o crescimento da vazão. Embora não se trate mais de um crescimento quase-linear como no primeiro intervalo, obtêm-se um crescimento considerável, mas com uma derivada menor. O fato desse crescimento não ser tão acentuado quanto no primeiro intervalo advém da necessidade de se realizar a fragmentação dos dados em vários quadros, aumentando assim o tempo de processamento para cada mensagem e reduzindo a vazão. Em torno de 3000 bytes espera-se um comportamento semelhante ao ocorrido ao redor de 1500 bytes, porém com um degrau menos acentuado.

Os gráficos nas figuras 5, 6, 7 e 8 mostram que quando se aumenta o número de processadores, o ganho percentual do protocolo LLC com relação ao TCP/IP se mantém. Nos gráficos 5 e 6 é possível observar que o comportamento dos protocolos é semelhante, independendo do número de processadores.

O gráfico 7 explicita uma tendência comum nos testes: a região para tamanhos de mensagens de 1400 a 1800 bytes mostra uma variação mais acentuada no ganho do protocolo LLC. A subida abrupta no ganho de desempenho, de 5% para 8%, ocorre de maneira mais suave nos gráficos das figuras 5, 6, e 8. O ponto de menor ganho nessa região é aquele para tamanho de mensagem de *1408* bytes, e o ponto com o máximo para o tamanho de *1472* bytes. Estes valores são os tamanhos do campo de dados de mensagens que podem ser transportadas pelos protocolos TCP/IP e LLC sem que haja fragmentação das mensagens.

É possível observar outro salto no ganho de desempenho do protocolo LLC que ocorre a partir do tamanho de mensagem *1792*. Esse tamanho de mensagem corresponde ao último tamanho comum testado na faixa de valores de tamanhos de mensagens nos quais a fragmentação e a remontagem dos pacotes agregam um custo elevado à transmissão dos dados.

[2]Incluindo dados de controle dos protocolos subjacentes.

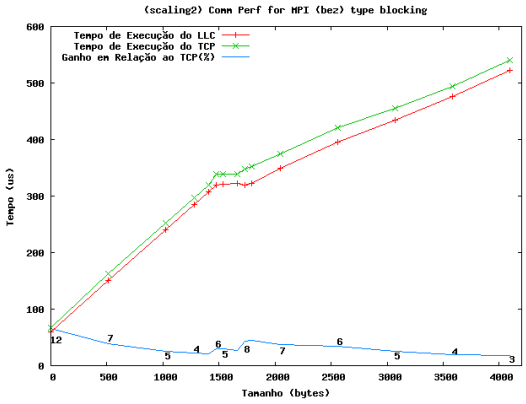

Figure 5: Execução do MPPTEST em 2 computadores

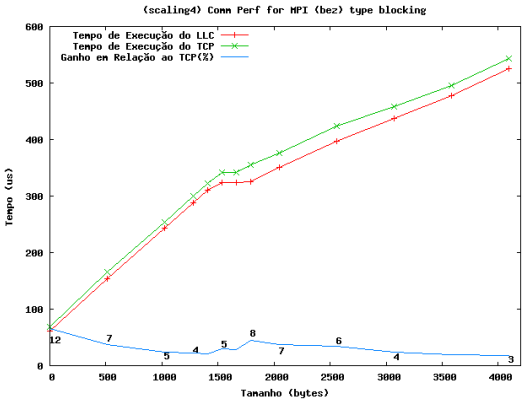

Figure 6: Execução do MPPTEST em 4 computadores

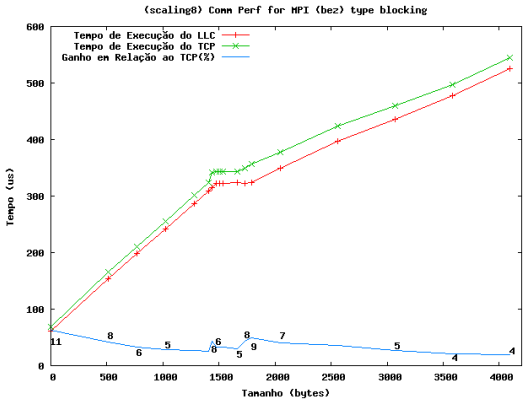

Figure 7: Execução do MPPTEST em 8 computadores

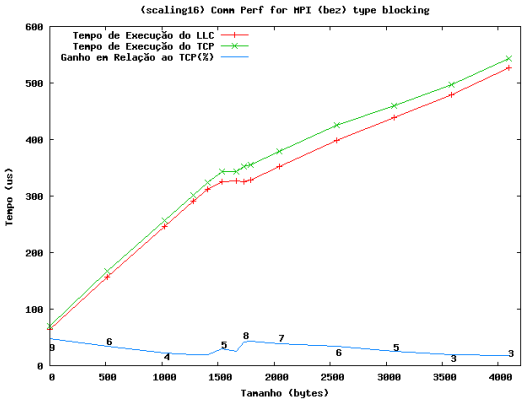

Figure 8: Execução do MPPTEST em 16 computadores

6 A Transformada de Fourier

As transformadas de Fourier [1] são utilizadas em várias aplicações científicas na física, na teoria dos números, combinatória, processamento de sinais, teoria das probabilidades, estatística, criptografia, acústica, oceanografia, ótica, geometria e outras áreas.

O *kernel* (núcleo) FFT (*Fast Fourier Transform*) utilizado nos testes é uma versão complexa e unidimensional do algoritmo descrito em [1].

Como o principal objetivo do presente estudo é avaliar a interferência dos protocolos das camadas inferiores (*LLC e TCP*), utilizou-se sempre a mesma massa de dados de entrada para o algoritmo; independentemente do número de processadores utilizados na resolução do algoritmo, a matriz de entrada teve sua ordem fixada em 20, por este ser o tamanho máximo de matriz suportado pela memória RAM dos microcomputadores que compõem o aglomerado.

O algoritmo foi modificado para calcular o tempo de processamento em cada um dos processadores em que a computação foi distribuída. Note que o tempo de comunicação do algoritmo FFT está diretamente relacionado ao tempo de processamento, sendo que o tempo de processamento aferido só é finalizado após a última troca de mensagens do

algoritmo, onde todos os nodos contém a matriz completa resolvida. Ao final da execução o programa retorna o tempo que cada processador dispendeu no cálculo do FFT. O valor utilizado para o cálculo dos resultados aqui apresentados é composto pela média aritmética dos tempos de execução dos processadores utilizados. Sendo assim, para dois processadores, têm-se como valor utilizado o resultado da média aritmética dos tempos de execução de cada um dos dois processadores. O valor médio foi utilizado porque a variância dos tempos de execução observada é muito pequena.

A figura 9 mostra o tempo de execução do algoritmo FFT para os dois protocolos TCP/IP e LLC, juntamente com o ganho do LLC em relação ao TCP. O eixo y mostra o tempo de execução em microsegundos e o eixo x, em escala logarítmica, mostra o número de processadores utilizados nos testes.

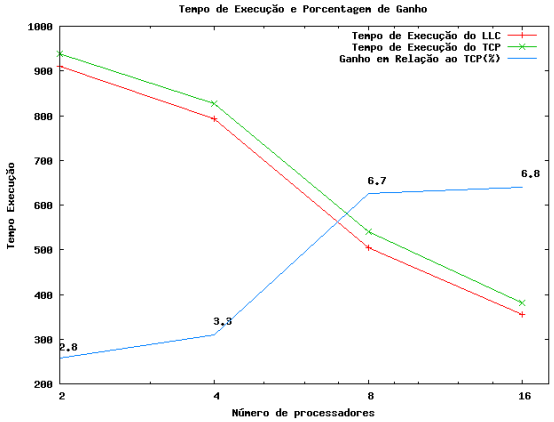

Figure 9: Tempo de Execução do Algoritmo FFT

Pode-se observar que à medida em que aumenta o número de processadores, para um conjunto fixo de dados (uma matriz de ordem 20), o tempo de execução do programa FFT decresce linearmente, e o ganho relativo do protocolo LLC aumenta consideravelmente. A redução no tempo de execução é esperada uma vez que ao dividir o problema em partes menores, cada processador resolve mais rápidamente sua parte, restando o tempo de agregar todos os resultados no resultado final.

O ganho obtido pelo protocolo LLC pode ser explicado da seguinte maneira. Para dois e quatro processadores, o ganho decorre da redução no tempo de comunicação porque o protocolo LLC é mais eficiente, e como os conjuntos de dados são relativamente grandes, há intensa comunicação entre os processadores.

Para oito e dezesseis processadores, os conjuntos de dados são relativamente menores, o número de mensagens trocadas é maior mas as mensagens são menores. Portanto, o ganho nestes dois casos decorre da maior eficiência do LLC com mensagens pequenas. O pequeno ganho de oito para dezesseis processadores decorre dos pequenos conjuntos de dados, sendo ultrapassado o ponto ótimo na relação computação/comunicação.

7 Ambiente de Execução dos Testes e Análise Estatística

Os ambiente de testes utilizado nesse estudo compõe-se de um aglomerado de computadores composto por 16 máquinas homogêneas com as seguintes características:

Hardware

- Processador: AMD Athlon 1.2 GHz
- Memória Cache L1 Instruções: 64KBytes
- Memória Cache L1 Dados: 64KBytes
- Memória Cache L2: 64KBytes
- Memória RAM: 128 MBytes
- Placa de Rede: Realtek 8139 séries C/C+ 100Mbits/s
- Rede: Ethernet
- Switch: DLINK DES-3226 10/100Mbits

Software

- Kernel Linux: 2.6.14-rc2-g8420e1b5 (modificado para suportar o socket LLC)
- OpenMPI: 1.0 stable + código BTL LLC escrito para esse estudo
- configurações específicas para o roteamento de pacotes LLC

Para a análise estatística dos dados amostrais adquiridos com a execução dos testes, utilizamos o teste da diferença entre duas médias utilizando as distribuições *t de Student*.

A hipótese nula (H_0) foi estabelecida como sendo a igualdade das duas médias. A hipótese alternativa (H_1) foi definida então como a diferença entre as duas médias.

Segundo [4, 2], uma maneira de obter uma forte evidência a partir dos dados com relação à afirmação feita pela hipótese testada é utilizar o nível de significância de 1%. Dessa maneira foi escolhido o nível de significância de 1% e o intervalo de confiança de 99% para a execução do teste *t de Student*.

Em [15] é dito que "quando a diferença entre as duas médias é testada com o uso de distribuições t, é necessária a suposição de que as variâncias das populações sejam iguais", o aplicativo estatístico R [20] permite estimar as variâncias independentemente uma da outra, sendo que nesses casos, é utilizada a modificação de Welch para os graus de liberdade do teste *t de Student*.

8 Conclusão

A utilização de aglomerados (*clusters*) de PCs, interligados por uma rede local tem suprido a crescente demanda por poder computacional em aplicações científicas. Para a programação de aplicações segundo o modelo de troca de mensagens em multicomputadores, são utilizadas bibliotecas de troca de mensagens, como MPI e PVM.

Os protocolos TCP/IP são utilizados como meio de transporte para a troca de mensagens utilizada nas bibliotecas de comunicação. Por conta da incerteza a respeito da rede na qual é empregado, o protocolo TCP incorpora uma série de mecanismos para garantir a entrega confiável das mensagens, bem como para manter a rede operando em condições razoáveis de tráfego e sem congestionamento.

Se a estrutura de rede na qual os dados trafegam for conhecida e restrita a uma rede local Ethernet, o que acontece em muitas instalações de aglomerados de computadores, o

protocolo de enlace da Ethernet (o padrão IEEE 802.2 – *Logical Link Control - LLC*) pode ser utilizado para substituir os protocolos TCP/IP. O protocolo LLC incorpora uma série de funcionalidades que são similares ou equivalentes a um sub-conjunto das funcionalidades do TCP, porém limitadas a uma rede local.

Por ser um protocolo mais simples e restrito que o protocolo TCP, é de se supor que um aglomerado de PCs construído sobre uma biblioteca que utilize LLC como seu protocolo de transporte seja mais eficiente —no ambiente de rede local Ethernet— do que um aglomerado baseado na mesma biblioteca mas com TCP como seu protocolo de transporte.

Este trabalho descreve uma adaptação da implementação da biblioteca MPI para executar com o protocolo LLC em uma rede local Ethernet e compara o desempenho desta implementação com uma implementação da MPI baseada em TCP/IP. Para a avaliação de desempenho foram utilizados os programas de teste NetPIPE e MPPTEST, além de dois núcleos de aplicativos *FFT*.

Nos testes executados utilizando o NetPIPE pode-se observar que, independente do tamanho da mensagem e para os valores testados, o protocolo LLC é mais eficiente que os protocolos TCP/IP. O maior ganho no desempenho do LLC ocorre na faixa de mensagens com tamanho até 13 bytes (100 bits), com ganho variando de 16 a 21% com relação ao TCP/IP. Embora o ganho percentual do LLC é menor com o aumento do tamanho da mensagem, esse ganho estabiliza-se na faixa de 3 a 5%, mesmo para mensagens de tamanho de 1500 bytes.

Os resultados do MPPTEST mostram que ao escalar o número de processadores utilizados no teste, o protocolo LLC tem ganho de desempenho que varia na faixa de 3 a 12%, ganho que varia na razão inversa ao tamanho das mensagens.

O mesmo comportamento é observado para os resultados da execução do programa da *Transformada Rápida de Fourier* (FFT), quando o LLC obtém ganhos que variam de 2.8% para 2 processadores a 6.8% para 16 processadores. O ganho de desempenho cresce com o número de processadores, quando o tamanho do conjunto de dados é fixo.

De tudo isso pode-se concluir que a utilização do protocolo LLC, no contexto de uma rede local Ethernet interligando um aglomerado de computadores, é válida para as aplicações descritas e pode ocasionar na redução do tempo de execução de outras aplicações que empregam a biblioteca MPI. Note-se também que o escopo de abrangência do protocolo LLC não se limita somente a uma rede pertencente a um aglomerado de computadores, mas toda rede local que empregue os protocolos IEEE *802.2* e *802.3*∗.

References

[1] David H. Bailey. FFTs in External or Hierarchical Memory. *Journal of Supercomputing*, 1(4):23–35, 1990.

[2] George E. P. Box, William G. Hunter, and J. Stuart Hunter. *Statistics for Experimenters An introduction to Design, Data Analysis, and Model Building.* John Wiley, 1st edition, 1976.

[3] A A Chien, M D Hill, and S S Mukherjee. Design challenges for high-performance network interfaces. *IEEE Computer*, 31(11):42–44, November 1998. `http://ftp.cs.wisc.edu/markhill/Papers/computer98_niguest.pdf`.

[4] David Roxbee Cox. *Planning of experiments.* John Wiley, 1992.

[5] David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus Erik Schauser, Eunice Santos, Ramesh Subramonian, and Thorsten von Eicken. LogP: Towards a Realistic Model of Parallel Computation. *4th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, May 1993. http://doi.acm.org/10.1145/155332.155333.

[6] David E Culler and Jaswinder Pal Singh. *Parallel Computer Architecture: a Hardware/Software Approach*. Morgan Kaufmann Publishers, Inc, 1999.

[7] Arnaldo Carvalho de Melo. TCPfying the poor cousins. *Linux Symposium*, 2(11):367–370, 2004. http://www.linuxsymposium.org/proceedings/reprints/Reprint-Melo-OLS2004.pdf.

[8] Sergio Luiz Marques Filho. Uma Abordagem Multithread em Aplicações Paralelas Usando MPI. Master's thesis, Universidade Federal do Paraná, Departamento de Informática, 2005.

[9] Message Passing Interface Forum. The message passing interface (mpi) standard, February 2006. http://www-unix.mcs.anl.gov/mpi/.

[10] Edgar Gabriel et al. Open MPI: Goals, concept, and design of a next generation MPI implementation. In *Proceedings, 11th European PVM/MPI Users' Group Meeting*, pages 97–104, September 2004.

[11] Richard L Graham, Timothy S Woodall, and Jeffrey M Squyres. OpenMPI: A flexible high performance MPI. In *Proc 6th Annual Int Conf on Parallel Processing and Applied Mathematics*, September 2005.

[12] William Gropp and Ewing L Lusk. Reproducible measurements of MPI performance characteristics, 1999. http://www.mcs.anl.gov/~gropp/bib/papers/1999/pvmmpi99/mpptest.pdf.

[13] Roberto A Hexsel. *A Quantitative Performance Evaluation of SCI Memory Hierarchies*. PhD dissertation, University of Edinburgh, Dept of Computer Science, October 1994. Tech Report CST-112-94. http://www.inf.ufpr.br/roberto/absMemHierAbstr.html.

[14] Vijay Karamcheti and Andrew A Chien. Software overhead in messaging layers: Where does the time go? *ASPLOS-VI*, 28, 29(11, 5):51–60, November 1994. http://doi.acm.org/10.1145/195473.195499.

[15] Leonard J. Kazmier. *Estatística Aplicada a Economia e Administração*. Mc Graw-Hill, 1982.

[16] Jeffrey C Mogul. TCP offload is a dumb idea whose time has come. *HotOS IX: The 9th Workshop on Hot Topics in Operating Systems*, April 2003. http://www.usenix.org/events/hotos03/tech/full_papers/mogul/mogul_html/index.html.

[17] OpenMPI Group. OpenMPI Frequently Asked Questions, April 2006. http://www.open-mpi.org/faq.

[18] J. Postel. Transmission Control Protocol. RFC 793 (Standard), September 1981. http://www.ietf.org/rfc/rfc793.txt.

[19] PVM Group. Parallel virtual machine, February 2006. http://www.csm.ornl.gov/pvm.

[20] R Project. The r project for statistical computing, January 2006. http://www.r-project.org/.

[21] Piyush Shivam and Jeffrey S Chase. On the elusive benefits of protocol offload. In *ACM SIGCOMM Workshop on Network-I/O convergence*, pages 179–184, 2003. http://doi.acm.org/10.1145/944747.944750.

[22] Quinn O Snell, Armin R Mikler, and John L Gustafson. NetPIPE: A network protocol independent performance evaluator, April 2006. http://www.scl.ameslab.gov/netpipe/paper/full.html.

[23] V S Sunderam. PVM: a framework for parallel distributed computing. *Concurrency, Practice and Experience*, 2(4):315–340, 1990. http://citeseer.ist.psu.edu/sunderam90pvm.html.

[24] Robbert van Renesse. Masking the overhead of protocol layering. In *SIGCOMM '96*, pages 96–104. ACM Press, 1996. http://doi.acm.org/10.1145/248156.248166.