Avaliação do Trafego de Rede no Sistema DSM *Nautilus* usando Atualizações: Relação entre o Número de Mensagens Trafegantes e o Tamanho Médio das Mensagens de Consistência

Mario Donato Marino[1], Geraldo Lino de Campos
{mario,geraldo}@regulus.pcs.usp.br

Departamento de Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo

RESUMO

O *Nautilus* é um sistema DSM (Distributed Shared Memory System) implementado em software que utiliza a Consistência de Liberação para minimizar o número de mensagens para a manutenção da consistência. Vai-se mostrar que, com a submissão de dois programas aplicativos diferentes, o SOR e o Mp3d (SPLASH I), na relação entre o número de mensagens trafegantes na rede, as mensagens de consistência predominam, assim como nos outros sistemas DSM existentes. Para completar a avaliação do tráfego de rede do DSM *Nautilus*[2], como as mensagens predominantes são as de consistência, é necessário saber seu tamanho médio, delineando, assim, o comportamento do tráfego. Os resultados do programa SOR mostraram que as mensagens de consistência predominam com 75% do total em relação às demais transmitidas, sendo o tamanho médio das mesmas em torno de 250 a 350 bytes. No caso do Mp3d, as mensagens de consistência predominam com 60 a 70% do total e seu tamanho médio girou em torno de 57 bytes.

ABSTRACT

Nautilus is a DSM Software (Distributed Shared Memory System) implemented in software using the Release Consistency, in order to minimize the number of messages to maintain the consistency. We show that submitting two applications programs, SOR and Mp3d (SPLASH I), the consistency messages prevails as we can see on the relation among the number of messages through the network, which can be observed in other DSM systems. Completing the evaluation of *Nautilus*, it's necessary to obtain the mean size of the consistency messages, delineating the traffic behaviour. The results of SOR show that the consistency messages prevails with 75% of the total among the other types of messages transmitted through the net, and its mean size is about 250 to 350 bytes. The resuts of Mp3d, show that the consistency messages prevails with 60 to 70% of the total and its mean size is about 57 bytes.

1) Introdução

[1] Suporte Financeiro do CNPq.
[2] O equipamento utilizado para avaliar o *Nautilus* é financiado pela Finep-Brasil.

A evolução crescente dos sistemas de inteconexão tem tornado a utilização do paradigma DSM (Distributed Shared Memory) [Stum90] cada vez mais no ambiente de supercomputação.

Um sistema DSM (*Distributed Shared Memory*) [Stum90] é uma abstração de um espaço de endereçamento compartilhado entre os nós de uma rede de computadores (distribuída). Pela adoção desta abstração, para o programador, é como se ele possuísse memória compartilhada, permitindo que programas paralelos escritos para máquinas de memória fisicamente compartilhada sejam facilmente portados. Essa abstração é conhecida como DSM (*Distributed Shared Memory*), como mostrado na figura 1, onde existem vários computadores (nós) interconectados por uma rede e que não possuem sua memória físicamente compartilhada.

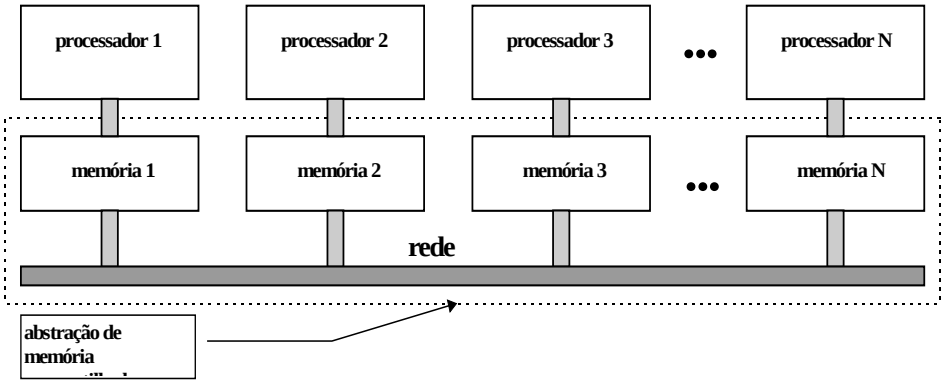

figura 1 - abstração de memória compartilhada

A adoção desta abstração [Stum90] implica em: (i) localizar e acessar os dados compartilhados; (ii) trocar e/ou replicar os dados ; (iii) manter a consistência dos dados entre os nós da rede; (iv) como foi dito, deve permitir a portabilidade com os programas paralelos escritos para memória compartilhada. Portanto, numa rede que esteja-se utilizando um sistema DSM, existem mensagens de: (i) páginas (dados),(ii) consistência e (iii) sincronização.

Como os programas paralelos exigem a sincronização para estabelecer uma coordenação entre os processos, é interessante usufruir um modelo de consistência de memória que garanta a consistência (seqüencial) nos pontos de sincronização do programa de modo a reduzir o número de mensagens para a manutenção da consistência. Os modelos que caem nestas características são os modelos de consistência de liberação (*release consistency*) que, portanto, **somente mantêm a consistência do programa aplicativo nos pontos de sincronização** (barreiras e semáforos).

Segundo Carter [Cart95], os sistemas de memória virtual compartilhada e distribuída (**DSM)** caracterizaram-se basicamente por duas fases distintas:

(i) por um elevado número de mensagens para a manutenção da consistência (por meio de mensagens de invalidação e atualização), utilizando principalmente a consistência seqüencial, como por exemplo o Ivy [Li86];

(ii) por uma redução drástica do número de mensagens para a manutenção da consistência por meio da adoção de técnicas como as múltiplas escritas concorrentes, mecanismo para cessar o envio de atualizações desnecessárias e principalmente a adoção da consistência de liberação que aproveita os pontos de sincronização natural dos

programas paralelos, barreiras e semáforos por exemplo, para manter o sistema consistente. São exemplos da segunda fase o Munin [Cart93], o Midway [Bers91, Bers93] (consistência de entrada), o TreadMarks [Kele95] (consistência de liberação preguiçosa) e o Quarks [Apwq95 ; Cart95] (domínio público).

O sistema DSM *Nautilus*, que se encaixa nesta segunda fase, apresenta as seguintes características: utiliza a consistência de liberação (*release consistency*) para reduzir o número de mensagens de consistência que trafegam na rede, faz uso de *threads* (ao invés de processos) para minimizar o chaveamento de contexto e pode ser executado em rede de PCs com Unix. Além disso, o *Nautilus* é compatível com o TreadMarks [Kele95] e com o Quarks [Apwq95; Cart95], bastando somente a troca das primitivas no programa aplicativo.

Com a submissão de dois programas aplicativos diferentes, o SOR e o Mp3d (SPLASH I), no sistema *Nautilus*, vai se poder verificar que as mensagens de consistência predominam sobre as demais que trafegam na rede (de sincronização e de páginas), assim como nos outros sistemas DSM existentes ([Kele95], Apwq]).

Como as mensagens de consistência são predominantes em número, para completar o estudo, vai-se mostrar o tamanho médio das mesmas. Assim, sabendo qual tipo de mensagem predomina na rede e seu respectivo tamanho, pode-se ter uma avaliação do tipo mais freqüente de mensagem e seu tamanho, podendo dimensionar ou propor um tipo de rede que melhor se adeqúe ao *Nautilus*.

Portanto o objetivo principal é avaliar qual o tipo de mensagem predominante que trafega na rede e seu tamanho médio, executando-se os apicativos SOR e Mp3d sobre o *Nautilus* numa rede de 12 PCs.

Os resultados do programa SOR mostraram que as mensagens de consistência predominam com 75% do total em relação às demais, obtendo-se o tamanho médio em torno de 250 a 350 bytes. No caso do Mp3d, as mensagens de consistência predominam com 60 % do total e seu tamanho médio girou em torno de 57 bytes.

2) Tipos de Mensagem que trafegam na rede de um sistema DSM

Como foi dito anteriormente, são vários os tipos de mensagens que trafegam num sistemas DSM: (i) mensagens contendo páginas, (ii) mensagens de sincronização e (iii) mensagens de consistência.

Resumidamente, as unidades de dados que um sistema DSM trabalha podem ser páginas e objetos, existindo, portanto, sistemas DSM que trabalham com páginas e outros com objetos. O Nautilus trabalha com páginas. Quanto um nó da rede não possui uma página ou quando ela está invalidada, um administrador de páginas emite uma mensagem a outro nó, para que ele envie a página procurada. Se este último a possuir, então uma mensagem contendo a página é emitida. Caso contrário uma mensagem de negação é emitida e o nó que está procurando a página deve tentar procurá-la em outro nó da rede.

Qualquer programa paralelo exige sincronização para a correta execução de seus processos constituintes. Portanto, numa rede de um sistema DSM as mensagens de sincronização vão ser emitidas para a sincronização dos processos que vão estar sendo executados em seus nós.

Quando os dados são replicados entre vários nós, é necessário que os nós se comuniquem para que os dados manipulados pelos mesmos estejam coerentes. Assim, são emitidas mensagens de consistência que irão trafegar na rede.

As consistência pode ser mantida por meio de mensagens de invalidação ou atualização.

Se um nó modifica uma página em uma seção crítica, as outras cópias da página que estão nos outros nós da rede, vão estar inválidas (não corretas). Assim, este nó emite mensagens de invalidação para os outros nós da rede. As mensagens de invalidação, quando recebidas por um nó, fazem com que a cópia da página (ou objeto) se torne inválida, isto é, como se o nó não possuísse mais a cópia da página. Se um nó fizer um acesso a uma página inválida, isto é, que ele não possui, conforme dito quando se falou em mensagens contendo páginas, um administrador é encarregado de procurar pela página necessária em outro nó da rede.

As mensagens de atualização carregam as informações atualizadas por cada nó entre duas sincronizações. Por exemplo, entre duas barreiras, uma página é modificada em um nó. Para a manutenção da consistência, as informações sofridas pela página até o momento da última sincronização devem ser propagadas aos outros nós para que o sistema fique com uma visão coerente. O agrupamento de todas as modificações sofridas por um conjunto de páginas entre dois instantes de sincronização constituem uma mensagem de consistência. A codificação das modificações sofridas por uma página é denominada de *diff* [Cart93]. Um *diff*, como contém as modificações sofridas por uma página, é bem menor em tamanho que uma página. Assim sendo, a transmissão de um *diff*, ao invés de página, mimiza a quantidade de dados que trafega na rede. Portanto, **uma mensagem de consistência de atualização contém vários diffs**.

3) Alguns Sistemas DSM

Vai-se mostrar agora como o *Nautilus* se encaixa entre os outros sistemas DSM.

O primeiro sistema DSM foi o Ivy [Li86]. Ele utiliza a consistência seqüencial, o que acarreta o envio de um grande número de mensagens para a manutenção da coerência.

O primeiro sistema DSM implementado em *software*, cujos programas apresentaram desempenho compatível com os mesmos programas convertidos para passagem de mensagens foi o Munin [Cart93]. Suas características principais são:

- primeiro sistema DSM (integrado à memória virtual), que conseguiu uma redução drástica do número de mensagens para a manutenção da consistência.
- consistência de liberação por meio do envio de mensagens de atualização;
- múltiplos protocolos de escrita, permitindo que mais de um nó escreva ao mesmo tempo em uma mesma página; as informações das modificações das páginas feitas em cada nó são armazenadas em estruturas conhecidas por *diffs* [Cart93]. Adotando esta técnica, foi o primeiro a eliminar o efeito do falso compartilhamento; portanto minimiza a quantidade de dados transmitida pela transmissão do *diff* e não de páginas;
- mecanismo de cessação do envio de mensagens de atualização que não serão utilizadas pelo nó que as recebe;
- suporta semáforos distribuídos e barreiras.
- opera em estações SUN 3/60;

O TreadMarks, primeiro DSM a conseguir desempenho próximo de uma máquina de memória fisicamente compartilhada [Kele95], apresenta as seguintes características:

- utiliza a consistência de liberação preguiçosa (*lazy release consistency*);

- pode utilizar protocolos de atualização, invalidação e híbrido;
- múltiplas escritas concorrentes, criação preguiçosa de *diffs*;
- transmite sempre *diffs* exceto a primeira busca (da página)
- implementado por uma biblioteca a nível de usuário;
- utiliza *sockets* e protocolo UDP para a minimização de *overheads*;
- pelo fato de utilizar a consistência de liberação preguiçosa, armazena uma grande quantidade de *diffs*, isto é muita memória implicando, em certas condições [Kele95] a se executar o processo de *garbage collection*;
- opera em estações SUN (com Unix 4.1.X) e Silicon Grafics (Irix);

O sistema *Nautilus*, seguindo a tendência internacional, apresenta as seguintes características:

- consistência de liberação como a do Munin [Cart93];
- implementado em forma de biblioteca a nível de usuário;
- tanto pode operar com invalidações, como também com atualizações e, neste caso, utiliza a técnica de múltiplos protocolos de escrita do Munin [Cart93];
- utiliza sockets e protocolo TCP ou UDP;
- utiliza *threads* na implementação para a minimização do chaveamento de contexto;
- opera em rede de PCs (Linux e FreeBSD 2.X) e de estações SUN (Unix Sun OS 4.1.X);

4) Avaliação

4.1) Programas de Avaliação

O Nautlius é submetido a dois programas de avaliação: SOR e Mp3d (SPLASH I).

O **SOR** (*Successive Over Relation*) é um programa que utiliza um algoritmo de relaxação, onde a partir de uma matriz de entrada bi-dimencional, para cada iteração do algoritmo, um elemento da matriz é atualizado em função dos elementos vizinhos.

O SOR utiliza somente barreiras para sincronização.

Com o objetivo de avaliar o comportamento do programa no *Nautilus*, o SOR foram executados com os tamanhos de matriz de entrada de 300x300, 400x400 e 600x600.

O **MP3D**, que faz parte do conjunto de aplicativos do SPLASH I [Singh91], simula um fluxo rarefeito sobre um objeto (veículos aerospaciais submetidos à atmosfera em velocidades hipersônicas) utilizando métodos de Monte Carlo (para simular as trajetórias de uma coleção de moléculas representativas).

O aplicativo **MP3D** utiliza barreiras e semáforos para a sincronização.

Com o objetivo de avaliar o comportamento do programa no DSM, o Mp3d foi executado sobre o Nautilus variando-se o número de moléculas, que é parâmetro de entrada. Foi executado para 1000, 2000 e 3000 moléculas.

4.2) Ambiente de Avaliação

Os aplicativos mensionados no item anterior são executados sobre o sistema *Nautilus* numa rede de 12 PCs, constituída por:

- nós: K6 233 MHz (AMD), 64 MB de memória e disco IDE de 2 GB;

- interconexão: placas de rede PCI padrão *fast-ethernet* (100 Mbits/s) com hub de 100 Mbits/s.

Para a obtenção das métricas deste estudo, a rede de interconexão acima mencionada foi totalmente isolada de qualquer rede externa.

O sistema operacional utilizado é o Linux Red Hat 5.0. Os aplicativos são executados e os resultados são extraídos com o *Nautilus* operando na rede de PCs.

5) Métricas de Avaliação

Para este estudo é importante salientar que o Nautilus está operando com o **protocolo TCP** e utilizando **mensagens de atualização** para a manutenção da consistência

O sistema Nautilus foi instrumentado de forma a medir os seguintes parâmetros: (i) o número de mensagens contendo pedidos de páginas, (ii) o número de mensagens contendo páginas, (iii) o número de mensagens de consistência, (iv) o número de mensagens de sincronização, (v) o número total de mensagens, (vi) a quantidade total em bytes de mensagens de consistência.

Para minimizar a quantidade de dados exibida, optou-se por mostrar em um gráfico a relação entre o número total das mensagens que trafegam na rede do Nautilus. Assim, serão exibidos dois gráficos (figura 2 e 3), mostrando as relações entre os números de mensagens que trafegam no sistema Nautilus para o SOR e para o Mp3d. Para obter as percentagens de mensagens de consistência, por exemplo, foi dividido o número de mensagens de consistência pelo número total de mensagens, isto é:

$$\text{percentagem de mensagens de consistência} = \frac{\text{(número de mensagens de consistência)}}{\text{(número total de mensagens)}}$$

Para a avaliação do tamanho médio das mensagens de consistência, fez-se o seguinte cálculo:

$$\text{tamanho médio das mensagens de consistência} = \frac{\text{(quantidade total em bytes de mensagens de consistêcia)}}{\text{(número de mensagens de consistência)}}$$

Para cada programa avaliado é calculado o tamanho médio, sendo exibidos na tabela 1.

6) Análise dos Resultados

Observando-se a legenda superior direita da figura 2, páginas pedidas são correspondentes a mensagens contendo pedidos de página e páginas enviadas são as mensagens contendo as páginas que foram pedidas por um nó.

Observando-se a figura 2, para os três tamanhos de matriz, percebe-se que as mensagens que predominam em número (cerca de 75 %) são de consistência, seguida pela de mensagens contendo páginas, resultado análogo aos outros sistemas DSM ([Kele95], [Apwq95], [Cart93]).

As mensagens de sincronização respondem pela segunda maior fatia com 20% do total de mensagens transmitidas.

Com uma fatia inferior 5% estão as mensagens contendo páginas e mensagens contendo pedidos por páginas.

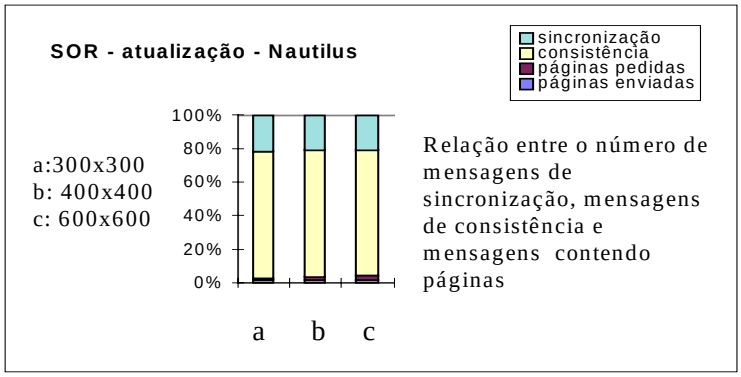

figura 2 - relação entre o número de mensagens de sincronização, consistência e páginas para o programa SOR

Para a figura 3, vale o mesmo comentário sobre a legenda feito para o SOR.

Observando-se a figura 3, percebe-se que as mensagens que predominam em número são de consistência (cerca de 60 % do total), para os três tamanhos de moléculas diferentes.

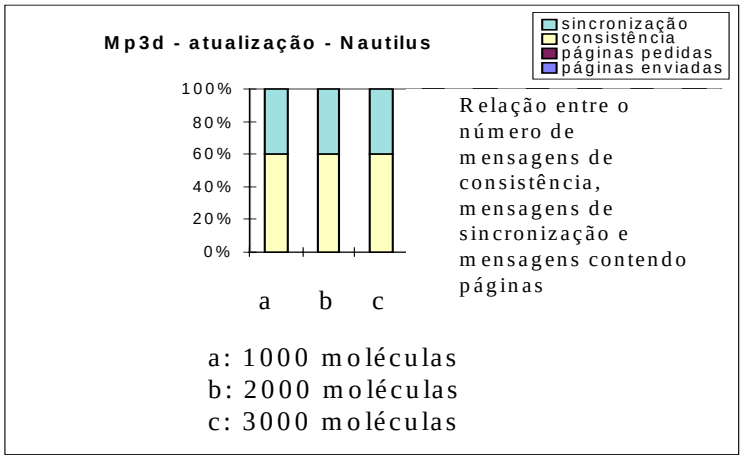

figura 3 - relação entre o número de mensagens de sincronização, consistência e páginas para o programa Mp3d

No caso do Mp3d, as mensagens de sincronização responderam por uma fatia maior que no SOR: 40%. Portanto este programa faz muito mais sincronização percentualmente que o SOR.

Praticamente, o número de mensagens contendo páginas e pedidos de páginas é nulo comparado com o número de mensagens de sincronização e de mensagens contendo páginas.

Como se pôde perceber neste estudo e nos estudos anteriores ([Kele95], [Cart93]), as mensagens de consistência são responsáveis pela maior parte do tráfego nas redes dos

sistemas DSM. Portanto justifica-se o esforço que vem sendo feito para minimizar o número de mensagens de consistência na rede.

A tabela 1 mostra o tamanho médio das mensagens de consistência para o SOR e para o Mp3d. Como se pode observar, em ambos os casos, o tamanho médio é inferior ao tamanho de página, que nos PCs é de 4 kilobytes e nos Alphas 8 kilobytes. Assim, trafegar mensagens de consistência, que contêm os *diffs*, ao invés de páginas, diminui drasticamente a quantidade de dados a ser transmitida.

programa avaliado	tamanho médio das mensagens de consistência
SOR	250 a 350
Mp3d	57

tabela 1: tamanho médio das mensagens de consistência para o SOR e Mp3d no Nautilus

6)Comentários, Conclusões e Trabalhos Futuros

Pode-se verificar neste estudo que, analogamente a outros sistemas DSM existentes, as mensagens que predominantemente trafegam na rede com o Nautilus em operação são mensagens de consistência.

Determinado seu tamanho e sua percentagem sobre o total, pode-se pensar em utilizar ou propor uma rede que se adeqúe melhor a tamanhos de mensagens de 57 a 350 bytes, isto é, uma rede cuja taxa de transferência seja alta para este tamanho de mensagem que trafega pela rede.

O estudo [Chiola] mostra que os protocolos TCP e UDP fornecem um *throughput* em torno de 1 MB/s, que é muito baixo, para tal tamanho de mensagem. Uma idéia para melhorar esse throughput seria utilizar deixar de usar os protocolos TCP e UDP e programar a placa de rede diretamente como propõe o estudo [Chiola], conseguindo melhorar o throughput para 6 a 8 MB/s.

Outro possível trabalho interessante seria utilizar uma rede ATM.

Também estão sendo estudados outros modelos de consistência como o a consistência de liberação preguiçosa [Kele95] e a consistência de escopo [Ifto96] para tentar minimizar o número de mensagens de consistência que trafegam na rede.

7**) Referências Bibliográficas**

[Apwq95] **Aplication programming with Quarks**, user manual, University of Utah, 1995.

[Bers91] Bershad N. B., Zekauskas M. J., **Midway: A Shared Memory Parallel Programming for Distributed Memory Multiprocessors**, Technical Report CMU-CS-91-170, Carnegie-Mellon University, Pittsburgh, September 1991.

[Bers93] Bershad B. N., Zekauskas M. J., Sawdon W. A., **The Midway Distributed Shared memory System**, COMPCON 1993.

[Cart93] Carter J. B., **Efficient Distributed Shared Memory Based on Multi-Protocol Release Consistency**, PHD Thesis, Rice University, Houston, Texas, September, 1993.

[Cart95] Carter J. B., Khandekar D., Kamb L., **Distributed Shared Memory: Where We are and Where We Should Be Headed**, Computer Systems Laboratory, Universisty of Utah, 1995.

[Ifto96] Iftode L., Singh J.P. amd Li K., Scope Consistency: A Bridge between Release Consistency and Entry Consistency, In the 8th Annual ACM Symposium on Prallel Algorithms and Architectures, 1996.

[Kele95] Keleher P., **Lazy Release Consistency for Distributed Shared Memory**, PHD Thesis, University of Rochester, Texas, Houston, January 1995.

[Li86] Li K, **Shared Virtual Memory on Loosely Coupled Multiprocessors**, PHD Thesis, Yale University, 1986.

[Sing91] Singh P. J., Weber W., Gupta A., Splash: Stanford parallel Applications for Shared-Memory. Computer Architercture News, Technical Report CSL-TR-91-469, Stanford University, April 1991.

[Stum90] Stumm M., Zhou S., **Algorithms Implementing Distributed Shared Memory, University of Toronto**, IEEE Computer, v. 23, n. 5, p. 54-64, May 1990.