Boletim Técnico da Escola Politécnica da USP
Departamento de Engenharia de Computação e
Sistemas Digitais

ISSN 1413-215X
BT/PCS/0114

Uma Avaliação do Sistema DSM Nautilus

Mario Donato Marino
Geraldo Lino de Campos

São Paulo - 2001

O presente trabalho é parte da tese de doutorado apresentada por Mario Donato Marino, sob orientação do Prof. Dr. Geraldo Lino de Campos.: "Uma Avaliação do Sistema DSM Nautilus", defendida em 22/02/2001, na EPUSP.

A íntegra da tese encontra-se à disposição com o autor e na Biblioteca de Engenharia de Eletricidade da Escola Politécnica da USP.

FICHA CATALOGRÁFICA

Marino, Mario Donato
Uma avaliação do sistema DSM Nautilus / M.D. Marino, G.L. de Campos. – São Paulo : EPUSP, 2001.
12 p. – (Boletim Técnico da Escola Politécnica da USP, Departamento de Engenharia de Computação e Sistemas Digitais, BT/PCS/0114)

1. Memória distribuída I. Campos, Geraldo Lino de II. Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais III. Título IV. Série
ISSN 1413-215X CDD 004.5

Uma Avaliação do Sistema DSM Nautilus

Mario Donato Marino

e-mail: mario@regulus.pcs.usp.br

Departamento de Computação e Sistemas Digitais

Engenharia de Eletricidade

Escola Politécnica da Universidade de São Paulo

1 Sistemas DSM

O modelo de memória compartilhada é uma extensão do modelo convencional com um único processador, pelo qual, qualquer mudança em um dado compartilhado de memória se torna imediatamente visível aos outros processadores porque a memória é fisicamente compatilhada entre os mesmos. O problema deste modelo é a alta contenção que os processadores provocam para manter a coerência de memória, apresentando assim uma grande perda de desempenho com um aumento da escala.

O modelo de memória distribuída não usa a abstração de uma única memória compartilhada no sistema, sendo que os processadores se comunicam por meio de passagem de mensagens. Se o modelo de passagem de mensagens for adotado, o programador de máquinas de memória fisicamente compartilhada vai ter de mudar seu estilo de programar, sendo obrigado a se preocupar com movimentação de dados, bem como seu empacotamento e linearização. Este modelo não apresenta o problema de contenção dos processadores pelo barramento[10]. Desta forma, para se ter a facilidade de programar do modelo de programação de memória compartilhada aliada a um bom desempenho, foi proposta a abstração de um espaço de endereçamento compartilhado entre processadores sobre um *cluster* de computadores, solução denominada de sistema *Distributed Shared Memory*[75] ou DSM, que pode ser visualizado na figura 1. Essa abstração permite ao usuário programar um *cluster* de computadores como se fosse uma máquina de memória fisicamente compartilhada.

Antes de explicar a constituição de um DSM, é conveniente detalhar o que são processos e *threads*[77]. Processos são entidades formadas por um espaço de endereçamento, um contador de programas, um *stack* e um conjunto de registradores. Um processo, também chamado de processo pesado, pode conter vários processos leves (*threads*), que possuem seus próprios contadores de programa e *stacks*, porém compartilham o mesmo espaço de endereçamento.

Pode-se dizer que um sistema DSM é formado vários conjunto de processos (ou de *threads*, se a implementação ou o sistema permitir), cada conjunto sendo executado em um dos nós da rede. Os processos (ou *threads*) pertencentes ao mesmo conjunto comunicam-se e os conjuntos de processos, que são executados em nós diferentes da rede, também se comunicam. O DSM deve transformar acessos à memória compartilhada em comunicação entre os conjuntos de processos[48].

Os DSMs, de forma tão transparente quanto possível, interceptam os acessos do usuário às memórias remotas e os transformam em mensagens apropriadas, que passam pela rede de interconexão. Portanto, o programador fica com a ilusão de estar com um grande espaço de endereçamento, pela eliminação explícita de troca de mensagens.

De forma análoga aos sistemas de memória compartilhada física, os DSMs podem replicar os dados, melhorando a localidade temporal e espacial. Da mesma forma que os sistemas de memória compartilhada física utilizam um bloco do *cache* para unidade de coerência, os DSMs podem também utilizar páginas de memória ou mesmo objetos como sendo sua unidade de compartilhamento. A grande unidade de compartilhamento, se comparada ao tamanho do bloco dos *caches* dos sistemas de memória compartilhada física, associada à latência dos *softwares* se combinam para constituir o grande desafio de projeto dos sistemas DSM[48].

Para melhorar seu desempenho, sistemas que apresentam memória compartilhada física utilizam *caches* locais, e portanto, replicam os dados nele contidos. Porém, ao utilizarem esses *caches*, devem manter sua coerência, já que os dados são replicados.

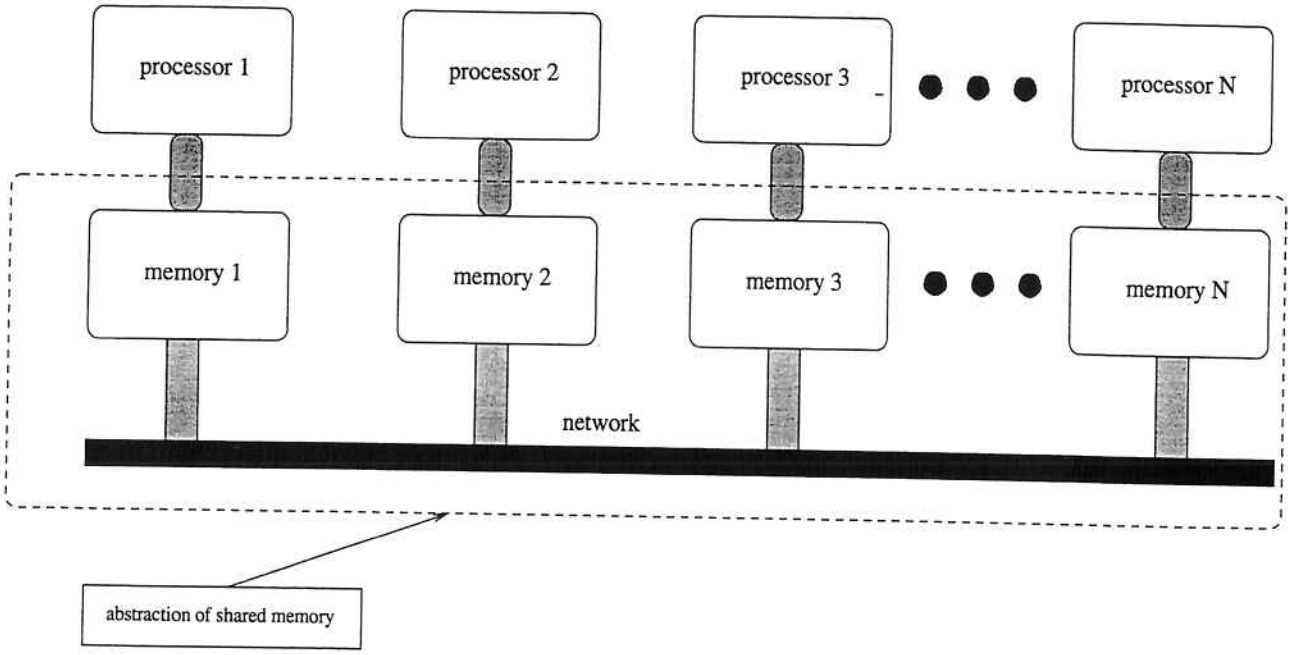

Figura 1: Abstração de DSM[10, 32]

Segundo Shi[66], a consistência e a coerência estão fortemente relacionadas e as condições de coerência consideram diferentes eventos de diferentes processadores para a mesma localidade, enquanto o modelo de consistência não somente considera os diferentes eventos na mesma localidade mas também impõe restrições na ordem dos eventos, isto é, na ordem de execução de cada processador. Assim uma combinação de coerência de *cache* e ordenação dos eventos irá determinar o comportamento global da memória do sistema. A seguinte relação lógica é igualmente proposta em [66]:

modelo de consistência de memória = protocolo de coerência + ordenação dos eventos em cada processador

Na definição acima proposta, o objetivo do protocolo de coerência é garantir a visão coerente da localidade de memória dos múltiplos processadores; a ordenação dos eventos descreve a ocorrência da seqüência dos eventos de memória emitidos através de cada processador. Em resumo, um modelo de consistência de memória especifica os requisitos de coerência num protocolo de coerência, isto é, qual a visão coerente da memória compartilhada que um protocolo de coerência deve prover[19]. Um protocolo de coerência de *cache* é de fato um mecanismo de propagação de dos valores recentemente escritos na memória de forma que os processadores devam ter uma nova visão coerente da mesma[19]. Para manter a coerência, dois mecanismos básicos são utilizados: a invalidação ou a atualização.

Um modelo de memória especifica quando as operações de coerência ocorrem e quais dados devem ser visíveis. Isto pode ser implementado com vários graus de *lazyness* na propagação e na aplicação [25]:

- das operações de coerência;
- das modificações de dados que devem ser aplicadas aos nós remotos por meio da emissão de mensagens.

O que se objetiva é maximizar a localidade, minimizando ao máximo a emissão de mensagens. É por isso que os pesquisadores investiram tanto nos modelos de consistência nos últimos anos .

O *modelo seqüencial* é o que apresenta consistência mais forte, o que implica em que todas as vezes que uma variável for modificada em um nó, as modificações dessa variável se tornam imediatamente visíveis aos outros nós (através do envio de mensagens). Esse imediatismo implica em um grande número de mensagens durante a execução do aplicativo sobre o DSM, causando um baixo desempenho[10].

Se a consistência seqüencial for mantida somente nos pontos de sincronização dos programas, o tráfego de mensagens de consistência será menor e, por conseguinte, o desempenho será melhor. Os modelos de consistência que se enquadram nestas características são os modelos de *consistência de liberação* (ou *release consistency*), que somente mantêm a consistência do programa aplicativo nos pontos de sincronização, em geral barreiras e semáforos[32].

Basicamente a evolução dos sistemas DSM não é nada mais do que a evolução da *lazyness* dos protocolos. Maior *lazyness* implica em[11]:

- comunicação menos freqüente e menos operações de protocolo de rede;
- maior dificuldade de programação e maior número de estados do protocolo de manutenção da coerência.

2 Resumo dos Objetivos dos Sistemas DSM

Segundo [15] apud [48], o desempenho dos DSMs depende de:

- velocidade de atendimento de um pedido à memória de um nó remoto;
- velocidade com que o sistema de memória virtual detecta uma falta de página;
- *overhead* do protocolo utilizado no DSM para atender uma falta de página;
- *overhead* do *software* de comunicação e seus vários níveis de envio;
- velocidade do meio de comunicação;
- número de mensagens contendo informações de consistência, dados e sincronização;
- sincronização de barreiras e semáforos;
- balanceamento de carga.

3 Proposta de uma nova classificação

Crê-se que a classificação de Carter é mais adequada do que a de Speight porque envolve somente os modelos de consistência e, pela sua adoção, a redução do número de mensagens de consistência pertinente. Portanto, foi proposta nos trabalhos[41, 42], uma extensão da classificação de Carter[10], mais adequada porque engloba modelos de consistência mais modernos. A classificação proposta é a seguinte:

1a. geração caracterizada por um grande número de mensagens para manter a coerência; um exemplo é o Ivy[37]

2a. geração caracterizada pela adoção de modelos de memória derivados do modelo de liberação (*release*) e redução do falso compartilhamento; os exemplos são o Munin[10] e o Quarks[10, 76];

3a. geração com modelos de consistência mais eficientes como o de liberação preguiçosa, de entrada e de escopo; os exemplos são o TreadMarks[32], o CVM[33], o JIAJIA[20] e o Nautilus[41].

Segue-se uma análise dos principais sistemas DSM.

4 Características principais do Nautilus

O Nautilus[39, 42] é um sistema DSM que apresenta as seguintes características:

- consistência de escopo (ou *scope consistency*[26]);
- técnica de múltiplas escritas concorrentes do Munin[10];
- minimização da criação de *diffs*[26, 32, 72, 20];
- primitivas compatíveis com o TreadMarks, Quarks e JIAJIA;
- implementado para rede de PCs com Linux 2.x;
- utiliza protocolo UDP para a minimização de *overheads*;
- *multithreading* para minimizar o chaveamento de contexto;
- não utiliza o sinal SIGIO, normalmente utilizado na implementação de DSMs para indicar a chegada de uma mensagem, diminuindo o número de chamadas de sistema e, desta forma, minimizando o *overhead*.

5 Contribuições do Nautilus

O Nautilus é o primeiro sistema DSM *multithreaded* implementado em plataforma Linux que utiliza o modelo de consistência de escopo. Esta frase é válida porque é o único DSM que simultaneamente apresenta estas características, já que:

- existem versões *multithreaded* do TreadMarks para Linux, porém ele utiliza o modelo de consistência de liberação preguiçosa, e não o modelo de consistência de escopo;
- o JIAJIA[20] é um DSM baseado na consistência de escopo porém não é *multithreaded*;
- o CVM[33] é um sistema DSM *multithreaded*, porém utiliza o modelo de consistência de liberação preguiçosa e não o modelo de consistência de escopo;
- o Brazos[72, 73] é um sistema DSM baseado em consistência de escopo, é *multithreaded*, porém somente opera em Windows NT, não em Linux, e assim, não pode ser adequadamente comparado.

Para melhorar o *speedup* das aplicações, duas técnicas são utilizadas pelo Nautilus:

- implementação *multithreaded*;
- *diffs* de páginas escritas pelo nó *home* não são criadas, pois estes nós já vão conter as páginas atualizadas.

A implementação *multithreaded* do Nautilus permite:

- a minimização de chaveamento de contexto;
- a não utilização do sinal SIGIO. A implementação *multithreaded* permite a eliminação do *overhead* do sinal SIGIO e a ativação do seu respectivo *handler* em todas as chegadas de mensagem.

A maior parte de sistemas DSMs baseados em páginas criados até hoje e implementados em plataformas Unix utilizam o sinal de SIGIO para ativar um *handler* que toma conta das mensagens que chegam da rede. Alguns exemplos de DSMs que se utilizam deste sinal são o TreadMarks, o Quarks e o JIAJIA. No Nautilus, um dos *threads* permanece bloqueado, quando tenta ler mensagens da rede. Enquanto bloqueado, permanece dormindo, não consumindo CPU. Esta técnica diminui o *overhead* do DSM permitindo que a CPU dê a maior prioridade possível ao programa do usuário. Portanto, o Nautilus é o primeiro sistema DSM, baseado em consistência escopo e em páginas, que não utiliza o SIGIO em sua implementação. A não utilização do sinal SIGIO minimiza a utilização da ativação de um *handler*, minimizando assim o *overhead*.

Da mesma forma que os outros sistemas, o Nautilus está preocupado em minimizar com os protocolos de rede. Para isto, o protocolo UDP é utilizado.

Como a idéia do Nautilus é aproveitar os programas utilizados pelos outros DSMs, preocupou-se com o aspecto compatibilidade. Como resultado, não existe nenhuma necessidade de rearranjo de código, quando se portam programas DSMs escritos com as primitivas do TreadMarks e do JIAJIA.

6 Avaliação Experimental

Neste item, serão avaliados alguns aplicativos, que serão executados em alguns DSMs e comparados com o Nautilus. Os parâmetros de comparação são os tempos de execução para 16 nós, t(16), speedups para 16 nós, Sp(16). número de mensagens, número de kbytes e número de mensagens de *diffs* e número de SIGSEGVs (sinal que indica violação de segmentação). Os DSMs que serão comparados com o Nautilus (Naut) serão o TreadMarks (Tmk) e o JIAJIA (JIA).

O ambiente de avaliação é constituído por uma rede de 16 PCs, interconectados por um switch 3C 3300. Cada PC é constituído de: i)Celeron 433 MHz; ii) 128 MB de memória SDRAM (66 MHz); iii) placa de rede fast-ethernet (100 Mbits/s) Intel Ether-Express Pro (conhecida também como Pro/100).

Os programas aplicativos que serão avaliados serão o LU (decomposição LU) do SPLASH-2, o SOR (equações de Laplace) da *Rice University* e o MM (multiplicação de matrizes). Os parâmetros de entrada dos programas estão localizados nas tabelas abaixo.

benchmarks	EP	EP	EP	LU	LU	LU	MM	MM	MM	SOR	SOR	SOR	Water	Barnes
parâmetros	M24	M26	M28	1024	1792	3072	1024	1792	3072	1024	1536	2048	1728	16384
t(1)	46.7	187.3	750.0	210.2	1148	5847	75.1	404.2	2063	2.9	6.3	11.4	8027	320
t(16).Tmk.4096	2.99	11.93	47.76	14.04	74.34	371.9	6.98	33.9	147.3	0.93	1.49	3.52	519.4	29
t(16).JIA.4096	3.13	12.04	47.74	14.83	76.82	380.0	10.3	37.2	169.2	2.18	4.61	10.45	546.0	72
t(16).Naut.4096.n	3.58	14.14	56.47	14.98	76.66	381.3	5.67	30.2	140.4	0.89	1.67	2.57	522.0	27
Sp(16).Tmk.4096	15.67	15.70	15.70	14.97	15.44	15.72	10.8	11.9	14.0	3.07	4.25	3.22	15.45	11.43
Sp(16).JIA.4096	14.95	15.56	15.71	14.18	14.95	15.39	7.29	10.9	12.2	1.31	1.37	1.09	14.70	4.44
Sp(16).Naut.4096.n	13.07	13.24	13.28	14.03	14.97	15.33	13.3	13.4	14.7	3.21	3.79	4.41	15.38	11.85

Tabela 1: Tempos e *speedups* para 16 nós

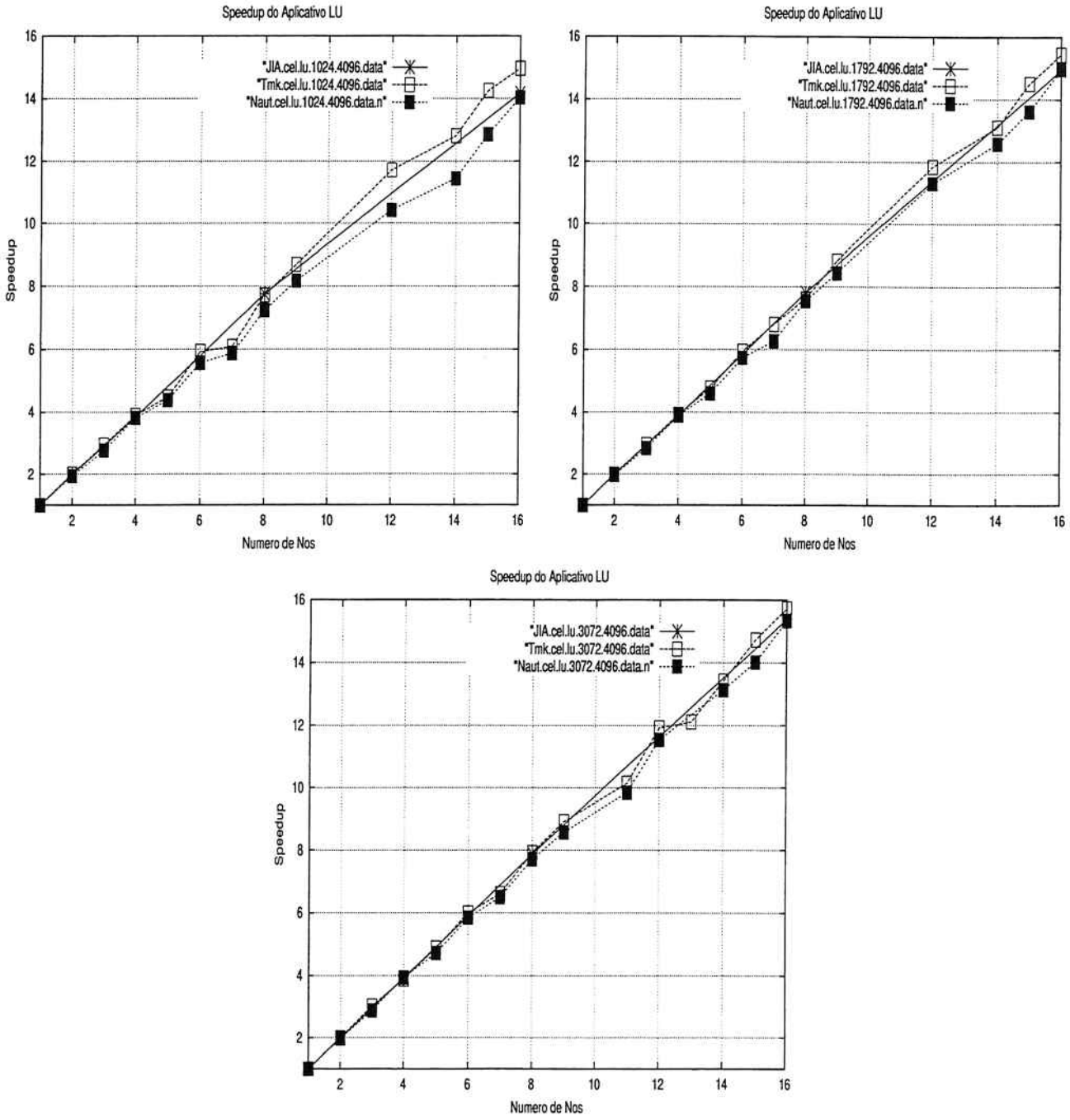

Figura 2: *Speedups* do aplicativo LU (SPLASH-2) - tamanhos 1024, 1792 e 3072

6.1 LU (SPLASH-2)

Os *speedups* do aplicativo LU podem ser vistos na figura 2, onde são mostrados os *speedups* dos três DSMs para três diferentes parâmetros de entrada 1024, 1792 e 3072 (figura 2). Os *speedups* aumentam com o aumento do tamanho do problema (parâmetro de entrada) pois a razão entre computação/comunicação do LU é do tipo $O(N^3/N^2)$[20].

Como se pode perceber pela figura 2, o comportamento geral dos gráficos é o seguinte: até 8 nós o JIAJIA é mais rápido que os outros, enquanto que para mais de 8 nós, o TreadMarks é mais rápido que o JIAJIA e este, mais rápido que o Nautilus, exceto para tamanho de matriz igual a 1792. Para 1792, a distribuição de dados desfavorece o Nautilus, o que acarreta uma perda de desempenho. Com 16 nós, o desempenho do Nautilus fica praticamente o mesmo do JIAJIA. Como se pode perceber na tabela 1, em valores numéricos, para 16 nós, a diferença de *speedup* entre o TreadMarks e o JIAJIA e entre TreadMarks e o Nautilus chega a respectivamente 5,3% e 6,3% para tamanho de matriz 1024, 3,2% e 3,0% para 1792 e, 2,1% e 2,5% para 3072. Deve-se observar também que à medida que se aumenta o tamanho das matrizes de entrada, mais altos ficam os *speedups* devido à melhora da localidade dos programas.

Outro ponto interessante que deve ser comentado é que o JIAJIA somente pode ser executado com número de nós potência de 2; portanto se alguns pontos dos gráficos de *speedup* do TreadMarks e Nautilus fossem desprezados não se perceberia o comportamento oscilatório das curvas, comportamento esse devido a divisão de blocos de matrizes em número de nós não múltiplos de potência de 2.

As razões que justificam o melhor *speedup* do TreadMarks em relação ao JIAJIA são: i)apesar do JIAJIA transmitir no global menos mensagens que o TreadMarks e menos kbytes, o JIAJIA transmite um maior número de mensagens relativas a páginas que o TreadMarks, que por sua vez transmite um número maior de *diffs*, graças à adoção dos modelos de consistência de escopo (*home-based*) e o modelo de consistência de liberação preguiçosa respectivamente; ii) sobrecarga dos nós por pedidos de página no JIAJIA, o que acaba aumentando o tempo de espera.

Pode-se dizer que o TreadMarks apresenta um maior desempenho em relação ao Nautilus devido predominantemente a maior utilização de mensagens relativas a páginas pelo Nautilus versus a utilização de *diffs* do TreadMarks, reflexo direto dos modelos de consistência adotados por cada um deles. A maior utilização de páginas pelo Nautilus acaba resultando numa maior quantidade de kbytes transmitida, acabando por ocupar uma banda maior. Uma observação complementar é que embora o JIAJIA e o Nautilus apresentem o mesmo modelo de consistência (escopo), diferem na distribuição de dados, conseqüentemente no falso compartilhamento, o que neste caso acabou prejudicando o desempenho do Nautilus. Não se deve esquecer também as diferentes codificações.

6.2 MM

Os *speedups* do aplicativo MM podem ser vistos na figura 3, onde são mostrados os *speedups* dos três DSMs para três diferentes parâmetros de entrada 1024, 1792 e 3072 (respectivamente figuras 3 a, b e c).

Como comportamento geral que pode ser percebido pela figura 3, o Nautilus é mais rápido que o TreadMarks e que o JIAJIA para os vários parâmetros de entrada (1024, 1792 e 3072). Como se pode perceber pela tabela 1, em valores numéricos, para 16 nós, a diferença de *speedup* entre o Nautilus e TreadMarks chega respectivamente a 18,8%, e entre Nautilus e JIAJIA a 45,0% para tamanho de matriz 1024, a 10,9%, e a 18,7% para 1792, a 4,8% e a 17,0% para 3072. Uma outra observação é que à medida que se aumenta o tamanho das matrizes de entrada, melhores ficam os *speedups* devido à melhora da localidade da multiplicação de matrizes (como comentado anteriormente, é um programa que tem uma relação comunicação/computação bem baixa, já que existem somente duas barreiras), e também mais próximas ficam as curvas de *speedup*.

Pode-se dizer que o Nautilus apresenta um melhor *speedup* em relação ao TreadMarks devido ao menor número global de mensagens transmitido, menor número de mensagens de páginas e também menor quantidade de kbytes. A distribuição de dados adotada pelo Nautilus, permitiu um tempo de *cold-startup* menor, desta forma resultando num número menor de mensagens relativas a páginas, conseqüentemente um menor número global de páginas, o que minimizou a utilização da rede, permitindo um melhor *speedup*. O *multithreading* e a não utilização do SIGIO também ajudaram a diminuir os *overheads* do Nautilus.

Confrontando o TreadMarks com o JIAJIA, apesar do JIAJIA transmitir menor número de mensagens, de kbytes e de páginas, o JIAJIA apresentou um *speedup* pior, devido a um possível problema de implementação.

6.3 SOR (Rice University)

Na figura 4, podem ser vistos os *speedups* do aplicativo SOR para os vários DSMs envolvendo diversos parâmetros de entrada (1024, 1536 e 2048). Pode-se perceber um aumento do *speedup* com o aumento da matriz de entrada pois, segundo Hu[20], a cada iteração existem $O(N^2)$ operações em ponto flutuante e O(N/tamanho da matriz) de faltas de página.

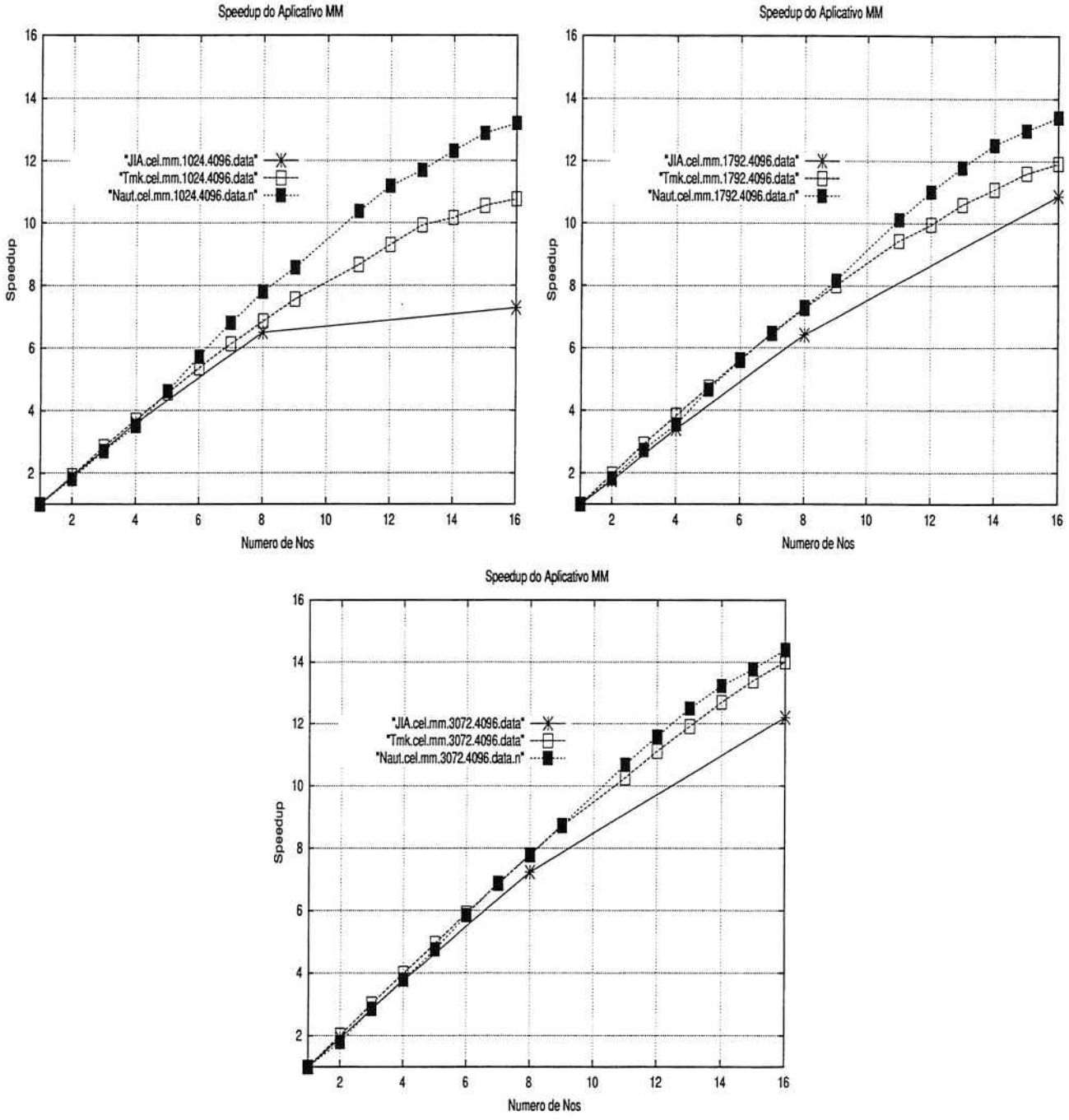

Figura 3: *Speedups* do aplicativo MM - tamanhos 1024, 1792 e 3072

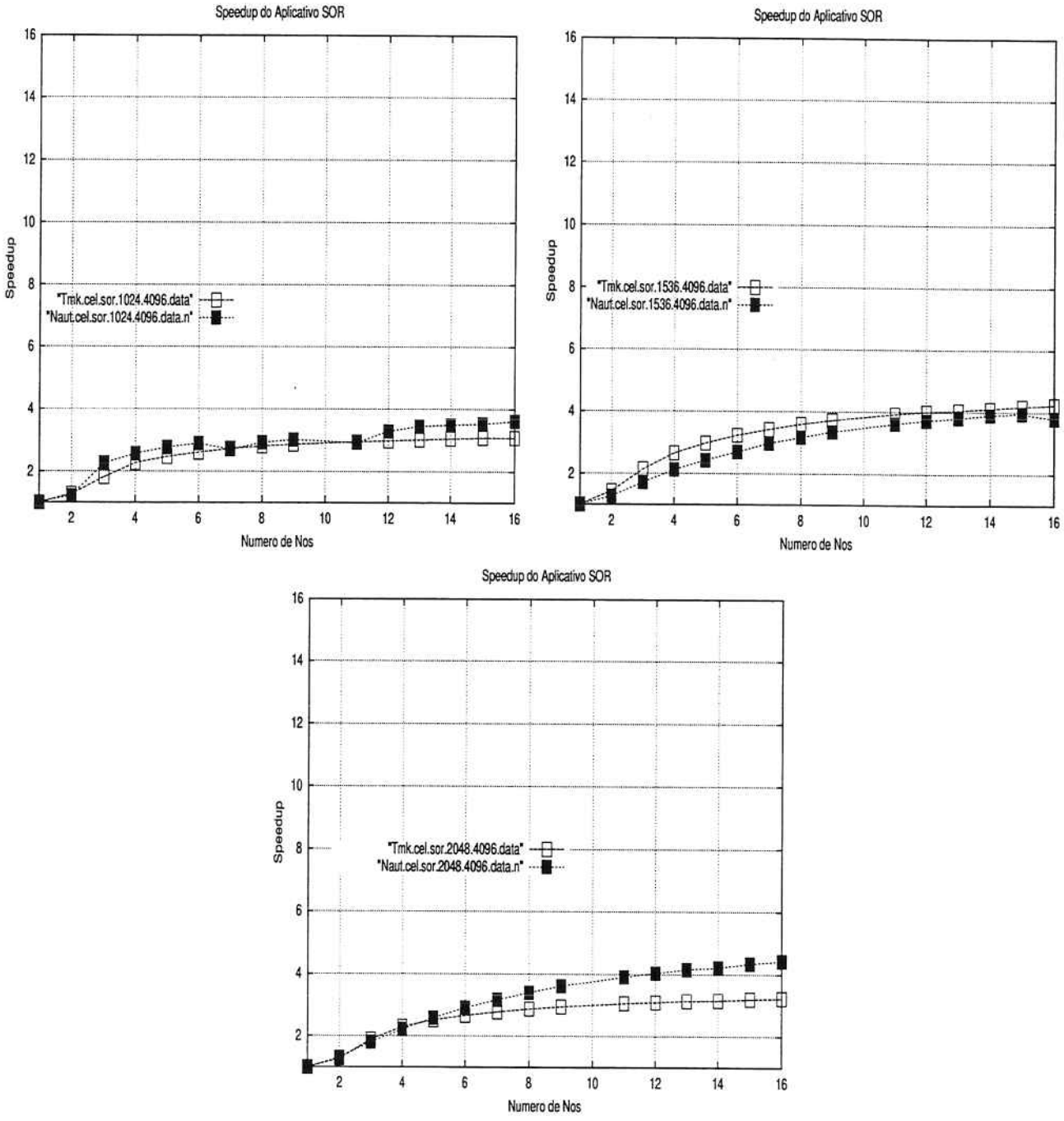

Figura 4: *Speedups* do aplicativo SOR (Rice-University) - tamanhos 1024, 1536 e 2048

Os *speedups* do JIAJIA não foram colocados na figura 4 porque não foram muito usuais, não sendo considerados para efeitos de comparação.

Pode-se dizer que o Nautilus é mais rápido que o JIAJIA para os tamanhos de matriz de 1024 e 2048, mas não para o caso de 1536. Pode-se justificar o melhor *speedup* graças ao menor número de mensagens transmitidas, menor número de kbytes, menor número de mensagens de páginas e menor número de mensagens de *diffs*. Estes parâmetros foram menores devido ao modelo de consistência adotado pelo Nautilus (de escopo), bem como a distribuição de dados que acaba minimizando o efeito do falso compartilhamento e minimizando o tempo de *cold-startup*. Para o caso de tamanho 1536, a distribuição de dados causa um maior efeito do falso compartilhamento o que prejudica o desempenho do Nautilus.

O *multithreading* e a não utilização do SIGIO também ajudaram a minimizar os *overheads* do Nautilus.

7 Conclusões Gerais

Nesta comparação, os DSMs TreadMarks, JIAJIA e Nautilus foram comparados em termos de tempo de execução, *speedups*, número de mensagens transmitidas na rede, número de kbytes transferidos, número de mensagens contendo páginas, número de SIGSEGVs e número de mensagens de *diffs*, para diversos programas aplicativos.

Para os programas EP e Water, os *speedups* dos DSMs empatam devido às características inerentes destes programas, respectivamente devido a pouca transmissão de dados do aplicativo EP e a alta sincronização do Water.

Para o aplicativo LU, o TreadMarks apresentou melhor desempenho em relação ao JIAJIA e Nautilus principalmente graças a seu modelo de consistência.

Para os programas MM, SOR e Barnes, o menor tempo de *cold-startup* e melhor distribuição de dados, auxiliados pelo modelo de consistência, permitiram ao Nautilus apresentar melhores *speedups* que os demais.

Referências

[1] *Accelerating the Standard for Speed*. Disponível em <http://www.gigabit-ethernet.org/index.html>. Acesso em outubro de 1999.

[2] *Active Messages*. Disponível em <http://now.cs.berkeley.edu/AM/active_messages.html>. Acesso em abril de 1998.

[3] *Adsmith DSM*. Disponível em <http://archi1.ee.ntu.edu.tw/~wyliang/adsmith>. Acesso em julho de 1998.

[4] *AMD Athlon Processor*. Estados Unidos da América. Disponível em <http://www.amd.com>. Acesso em dezembro de 1999.

[5] AMZA C.; COX A. L.; SWARKADAS S. JIN L. J.; RAMAJANI K.; ZWAENEPOEL W. Adaptive Protocols for Software Distributed Shared Memory. Artigo em revista. Proceedings of IEEE, Special Issue on Distributed Shared Memory, pp. 467-475, março, 1999.

[6] BERSHAD B. N.; ZEKAUSKAS M. J.; SAWDON W. A. The Midway Distributed Shared Memory System. Artigo de revista. COMPCOM, 1993.

[7] BUONADONNA P. *Unet Berkeley Project*. Universidade da Califórnia, Berkeley, Estados Unidos da América. Disponível em <http://www2.cs.cornell.edu/U-Net/Default.html>. Acesso em junho de 1998.

[8] *BUONADONNA P. Berkeley VIA Project, An Investigation of the Virtual Interface Architecture*. Universidade da Califórnia, Berkeley, Estados Unidos da América, 1998. Disponível em <http://www.cs.berkeley.edu/~philipb/via>. Acesso em setembro de 1998.

[9] CARTER J. B.; KHANDEKAR D.; KAMB L. *Distributed Shared Memory: Where We are and Where we Should Headed*. Technical Report. Computer Systems Laboratory, Universidade de Utah, 1995.

[10] CARTER J. B.; *Efficient Distributed Shared Memory Based on Multi-protocol Release Consistency*. Tese de doutorado. Universidade de Rice , Houston, Texas, setembro, 1993.

[11] CARTER J. B.; BENNETT J. K., ZWAENEPOEL W. Techniques for Reducing Consistency-Related Communication in Distributed Shared Memory System. Artigo de Revista. ACM Transactions on Computer Systems, Vol. 13, no. 3, pp. 205-244, agosto, 1995.

[12] *Cashemere Project*. Universidade de Rochester. Nova Iorque. Estados Unidos da América. Disponível em <http://www.rochester.edu/research/cashemere>. Acesso em outubro de 1999.

[13] CIACCIO G. *GAMMA Project: Genoa Active Message MAchine*. Disponível em <http://www.disi.unige.it/project/gamma/>. Acesso em fevereiro de 1998.

[14] COX A.L.; LARA E.; HU Y.C.; ZWAENEPOEL W. *A Performance Comparison of Homeless and Home-based Lazy Release Consistency Protocols in Software Shared Memory*. Artigo em evento. Proceedings of the Fifth High Performance Computer Architecture Conference (HPCAC), janeiro, 1999.

[15] *Distributed Inter-Process Communication (DIPC)* System. Disponível em <http://wallybox.cei.net/dipc>. Acesso em maio de 1999.

[16] ESKICIOGLU M. R.; MARSLAND T. A.; HU W.; SHI W. *Evaluation of JIAJIA Software DSM System on High Performance Architectures*. Artigo de evento. Thirty-Second Hawaii International Conference on System Sciences(HICSS-32), Havaí, Estados Unidos da América, janeiro, 1999.

[17] *Fast-messages* package. Estados Unidos da América. Disponível em <http://www_csag.uscd.edu/projects/comm/fm.html>. Acesso em setembro de 1999.

[18] HU W.; SHI W.; TANGZ. *Adaptive Write Detection in Home-based Software DSMs*. Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing, Redondo Beach, Califórnia, Estados Unidos da América, agosto, 1999.

[19] HU W.; SHI W.; TANG Z.; LI M., A Lock-based Cache Coherence for Scope Consistency. Artigo em revista. Journal of Computer Science and Technology, Vol 13, No. 2, pp. 97-110, 1998.

[20] HU W.; SHI W.; TANG Z. *JIAJIA: An SVM System Based on A New Cache Coherence Protocol*. Artigo em evento. Proceedings of the High Performance Computing and Networking (HPCN'99), LNCS 1593, pp. 463-472, Springer, Amsterdam, Países Baixos, abril, 1999.

[21] HU W.; SHI W.; TANG Z. *Home Migration in Home-Based Software DSMs*. Artigo em evento. 1st Workshop on Software Distributed Shared Memory (WSDSM '99), Rhodes, Grécia, 25 de junho, 1999.

[22] HU W.; SHI W.; TANG Z. *Reducing System Overheads in Home-based Software DSMs*. Artigo em evento. 13th International Parallel Processing Symposium & 10th Symposium on Parallel and Distributed Processing(IPPS/SPDP'99), IEEE CS Press, pp. 167-173, San Juan, Puerto Rico, abril, 1999.

[23] HU W; SHI W.; TANG Z. *Write Detection in Home-based Software DSMs*. Artigo em evento. Proceedings of Euro-Par'99, , Tolouse, France, janeiro, August 31-September 2, 1999.

[24] IFTODE L. *Home-based Shared Virtual Memory*. Tese de doutorado. Universidade de Princeton , Estados Unidos da América, agosto, 1998.

[25] IFTODE L; SINGH J. P.; Iftode L., Singh J. P. Shared Virtual Memory: Progress and Challenges. Artigo de revista. Proceedings of the IEEE, Vol 87, No. 3, março, 1999.

[26] IFTODE L.; SINGH J. P.; LI K. *Scope Consistency: A bridge between release consistency and entry consistency*. Artigo em evento. Proceedings of the 8th ACM Annual Symposium on Parallel Algorithms and Architectures (SPAA'96), pp. 277-287, junho, 1996.

[27] IFTODE L.; SINGH J.P.; LI K. *Understanding Application Performance on Shared Virtual Memory*. Artigo de evento. Proceedings of the 23rd Annual International Symposium on Computer Architecture, maio, 1996.

[28] JIANG D.; SHAN H.; SINGH J. P. *Application Restructuring and Performance Portability on Shared Virtual Memory and Hardware-Coherent Multiprocessors*. Artigo em evento. Proceedings of the 6th ACM Symposium on Principles and Practice of Parallel Programming, junho, 1997.

[29] JOHNSON D. B.; ZWAENEPOEL W. The Peregrine High-Performance RPC System. Artigo em revista. Software: Practice and Experience, Vol. 23, No. 2, pp. 201-221, fevereiro, 1993.

[30] KARLSSON S.;BRORSSON M. *An Infrastructure for Portable and Efficient Software DSM*. Artigo em evento. 1st Workshop on Software Distributed Shared Memory (WSDSM '99), Rhodes, Grécia, 25 de junho, 1999.

[31] KELEHER P. *Distributed Shared Memory Home Pages*. Estados Unidos. Universidade de Maryland. Disponível em <http://www.cs.umd.edu/~keleher/dsm.html>. Acesso em setembro de 1997.

[32] KELEHER P. *Lazy Release Consistency for Distributed Shared Memory*. Tese de doutorado. Universidade de Rochester, Texas, Houston, janeiro, 1995.

[33] KELEHER P. *The Relative Importance of Concurrent Writers and Weak Consistency Models*. Artigo de evento. Proceedings of the 16th International Conference on Distributed Computing Systems (ICDCS-16), pp. 91-98, maio, 1996.

[34] KELEHER P. Update Protocols and Cluster-based Shared Memory. Artigo de revista. In Computer Communications, 22(11), pp. 1045-1055, julho, 1999.

[35] KELEHER P. *Update Protocols and Iterative Scientific Applications*. Artigo de evento. The 12th International Parallel Processing Symposium, março, 1998.

[36] *Larchant Project*. INRIA. França. Disponível em <http://www-sor.inria.fr/SOR/projects/larchant.html>. Acesso em setembro de 1999.

[37] LI K. *Shared Virtual Memory on Loosely Coupled Multiprocessors*, tese de doutorado, Yale University, 1986.

[38] LU H.; DWARKADAS S.; COX A. L.; ZWAENEPOEL W. *Quantifying the Performance Differences between PVM and TreadMarks*. Artigo em revista. Journal of Parallel and Distributed Computation, Vol. 43, No. 2, pp. 65-78, junho, 1997.

[39] MARINO M. D.; CAMPOS G.L. *A DSM Speedup Comparison: TreadMarks, JIAJIA and Nautilus*. Artigo em evento. Proceedings of the International Conference on Parallell and Distributed Processing Techniques and Applications (PDPTA), Vol. IV, pp. 1744-1748, Las Vegas, Nevada, Estados Unidos da América, julho, 1999.

[40] MARINO M. D.; CAMPOS G. L. *An Evaluation of Page Aggregation Technique on Different DSM Systems*. Third International Symposium on High Performance Computing (ISHPC2K), LLNCS 1940, Springer Verlag, v1, pp134-145, Tokio, Japão, outubro, 2000.

[41] MARINO M.D.; CAMPOS G. L.; SATO L.M. *An Evaluation of the Speedup of Nautilus DSM System*, Parallel Symposyum(IASTED PDCS), Boston, proceedings em CD, novembro, 1999.

[42] MARINO M. D.;, CAMPOS G.L. *Nautilus: A Three Third Generation DSM System*, 8th Workshop on Shared Memory Multiprocessors in conjunction with ISCA99, Atlanta, Georgia, Estados Unidos da América, abril/maio, 1999.

[43] MARINO M. D.; CAMPOS G.L. *A Preliminary Comparison Between Two Scope Consistency DSM Systems: JIAJIA and Nautilus*. Artigo em evento. Proceedings of the International Workshop on Parallel Computing at Seventh International Conference on Parallel Processing., IEEE proceedings, v1., pp319-324, Fukujima, Japão, julho, 1999.

[44] MARINO M.D.; CAMPOS G. L. A Preliminary DSM Speedup Comparison: JIAJIA x Nautilus. Capítulo de livro. Kluwer Academics, HPCS99, Kingston, Canada, junho, 1999.

[45] MARINO M. D.; CAMPOS G. L. *A Preliminary Study of Cache-Only Write Detection Technique for Nautilus DSM*. Simpósio Brasileiro de Arquitetura de Computadores e Processamento de Alto Desempenho (SBAC-PAD), v1, pp 217-224, Águas de São Pedro, São Paulo, Brasil, outubro, 2000.

[46] MARINO M. D.; CAMPOS G.L. *Techniques for Improving the Speedup of Nautilus DSM*. Artigo em evento. Proceedings of the International Conference on Parallell and Distributed Processing Techniques and Applications (PDPTA), proceedings em CD, CSREA, Las Vegas, Estados Unidos da América, junho, 2000.

[47] MARINO M. D.; CAMPOS G. L. *The Speedup of Nautilus, a new DSM System, Compared to Treadmarks*. Second International Conference on Parallel and Computing Systems (PCS99), v1. p70-78, Baja California, México, agosto, 1999.

[48] MARINO, M.D. *Pulsar: Um Sistema de Memória Distribuída e Compartilhada baseado em Consistência de Memória Relaxada*, Dissertação de Mestrado, Escola Politécnica da Universidade de São Paulo, outubro, 1996.

[49] *Mirage Project*. Disponível em <http://cs.ucr.edu/projects/mirage.html>. Acesso em novembro de 1999.

[50] MONNERAT L.R.; BIANCHINI R. *Efficiently Adapting to Sharing Patterns in Software DSMs*. Artigo em evento. Proceedings of the 4th IEEE International Symposium on High-Performance Computer Architecture (HPCA98), fevereiro, 1998.

[51] *Mosix Scalable Computing for Linux*. Instituto de Ciências da Computação, Universidade de Hebrew, Jerusalem, Israel. Disponível em <http://www.mosix.cs.huji.ac.il>. Acesso em maio de 1999.

[52] MOWRY T. C.; Cnah Q. C.; LO A. K. W. *Comparative Evaluation of Latency Tolerance Techniques for Software Distributed Shared Memory*. Artigo em evento. Proceedings of the 4th IEEE Symposium on High Performance Computer Architechture, 1998.

[53] *MPICH - A Portable Implementation of MPI*. Laboratório Nacional de Argone. Estados Unidos da América. Disponível em <http://www-unix.mcs.anl.gov/mpi/mpich>. Acesso em agosto de 1999.

[54] *Myrinet Overview*. Disponível em <http://www.myri.com>. Acesso em novembro de 1997.

[55] NG M.C.; WONG W.F. *Adaptive schemes for home-based DSM systems*. Artigo em evento. 1st Workshop on Software Distributed Shared Memory (WSDSM '99) Rhodes, Grécia, 25 de junho, 1999.

[56] N. P.; MANOJ R. *CAS-DSM: A compiler-assisted DSM*. Artigo em evento. 1st Workshop on Software Distributed Shared Memory (WSDSM '99) Rhodes, Grécia, 25 de junho, 1999.

[57] *OpenMP Resources*. Disponível em <www.openmp.org/index.cgi?resources>. Acesso em novembro de 1999.

[58] *Parallel I/O Archive* (PARIO). Faculdade Dartmouth, Departamento de Computação. Estados Unidos da América. Disponível em <http://www.cs.dartmouth.edu/pario>. Acesso em agosto de 1999.

[59] *Parmon*. Disponível em <http://www.dgs.monash.edu.au/~rajkumar/papers/parmon.html>. Acesso em novembro de 1998.

[60] *PVM - Parallel Virtual Machine*. Laboratório Nacional de Oak Ridge. Estados Unidos da América. Disponível em <http://www.epm.ornl.gov/pvm>. Acesso em agosto de 1999.

[61] RAJKMAR B. *High Performance Cluster Computing: Arquitectures and Systems*. Austrália. Disponível em <http://www.dgs.monash.edu.au/~rajkumar/cluster/index.html>. Acesso em novembro de 1998.

[62] SAMANTA R.; BILAS A.; IFTODE L.; SINGH J. P. *Home-based SVM protocols for SMP clusters: Design and Performance*. Artigo de evento. In Proceedings of the 4th International Symposium on High-Performance Computer Architecture, fevereiro, 1998.

[63] SATO L. M.; MIDORIKAWA E. T.; BERNAL V. B. *Práticas em Programação Paralela*. Tutorial. Simpósio Brasileiro de Arquitetura de Computadores e Processamento de Alto Desempenho (SBAC-PAD), v1, pp. 1-38, São Paulo, 1992.

[64] SCALES D. J.; GHARACHORLOO K.; THEKKATH C. *Shasta: A Low Overhead, Software Only Approach for Supporting Fine-Grain Shared Memory*, Artigo em evento. Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 297-306, outubro, 1996.

[65] *Shared Virtual Memory Library (SVMlib)*. Disponível em <http://www.lfbs.rwth-aachen.de/~sven/SVMlib>. Acesso em janeiro de 1999.

[66] SHI W.; HU W.; TANG Z. An Interaction of Cache Coherence Protocol and Memory Consistency Model. Artigo em revista. ACM Operating Systems Review, Vol. 31, No. 4, pp. 41-54, outubro, 1997.

[67] SHI W.; MAO Y.; TANG Z. *Communication Substrate for Software DSMs*. Artigo em evento. Proceedings of the 11th IASTED International Conference on Parallel and Distributed Computing and Systems, , MIT, Boston, Estados Unidos da América, novembro (3-6), 1999

[68] SHI W., MA J.; TANG Z. *High Efficient Parallel Computation of Resonant Frequencies of Waveguided Loaded Cavetives on JIAJIA Software DSM System*. Artigo em evento. In Proceedings of the High Performance Computing and Networking (HPCN'99), LNCS 1593, pp. 1147-1150, Springer, Amsterdam, Países Baixos, abril, 1999.

[69] SHI W.; HU W.; TANG Z. Where Does the Time Go in SVM System?-Experiences with JIAJIA. Artigo em revista. Journal of Computer Science and Technology, Vol 14, No.3, pp.193-205, maio, 1999.

[70] *Shrimp Princeton Project*. Universidade de Princeton. New Jersey. Estados Unidos da América. Disponível em <http:/www.cs.Princeton.EDU:80/shrimp>. Acesso em julho de 1999.

[71] *Simple Coma*. Disponível em <http://playground.Sun.COM:80/pub/S3.mp/simple-coma/>. Acesso em abril de 1999.

[72] SPEIGHT E.; BENNETT J. K. *Brazos: A third generation DSM system*. Artigo em evento. Proceedings of the 1997 USENIX Windows/NT Workshop, pp. 95-106, agosto, 1997.

[73] SPEIGHT E.; BENNETT J. K. *Using Multicast and Multithreading to Reduce Communication in Software DSM Systems*,. Artigo em evento. Proceedings of the Fourth Symposyum on High Performance Architecture (HPCA), pp. 312-323, fevereiro, 1998.

[74] STAFFORD W. *The Berkeley Now Project*. Estados Unidos da América. Disponível em <http://now.cs.berkeley.edu>. Acesso em janeiro de 1998.

[75] STUM M; ZHOU S. *Algorithms Implementing Distributed Shared Memory*, artigo de revista, IEEE Computer v.23 , n.5 , pp.54-64, maio, 1990.

[76] SWANSON M.; STOLLER L.; CARTER J. *Making Distributed Shared Memory Simple, Yet Efficient*. Technical report. Laboratório de Sistemas de Computação, Universidade de Utah, 1998.

[77] TANEMBAUM A.S., *Distributed Operating Systems*. Livro. Primeira Edição, Prentice Hall, 1995.

[78] *Task Force on Cluster Computing* (IEEE-TFCC). Disponível em <http://www.ieeetfcc.org>. Acesso em janeiro de 2001.

[79] *The Beowulf Underground*. Estados Unidos da América. Disponível em <www.beowulf-underground.org/documentation.html>. Acesso em setembro de 1997.

[80] THITIKAMOL K.; KELEHER P. *Multi-Threading and Remote Latency in Software DSMs*. Artigo em evento. The 17th International Conference on Distributed Computing Systems, maio, 1997.

[81] UENG J. C.; SHIEH C. K.;LIN Q.C. *Design and Implementation of Proteus*. Artigo em evento. 1st Workshop on Software Distributed Shared Memory (WSDSM '99), Rhodes, Grécia, 25 de junho, 1999.

[82] *Vote-Communicatiion Support System*. Instituto de Arquitetura de Computadores e Tecnologia de Software. Berlin. Alemanha. Disponível em <http://www.first.gmd.de/vote >. Acesso em outubro de 1999.

[83] WEIWU H. *Reducing Message Overhead in Home Based Software DSMs*. Artigo em evento. Proceedings of ACM 1st Workshop on Software DSM System, Grécia, junho, 1999.

[84] WELSH M. ;EICKEN T. V.; Low-Latency Commnunication over Fast-Ethernet, Technical Report, Universidade de Berkeley, 1998.

[85] *Wisconsin Wind Tunnel Project*. Universidade de Wisconsin. Estados Unidos da América. Disponível em <http://www.cs.wisc.edu/~wwt>. Acesso em dezembro de 1997.

[86] WOO S.; OHARA M.; TORRIE E.; SINGH J. P.; GUPTA A. *The SPLASH-2 programs: Characterization and methodological considerations*. artigo de revista, *In Proceedings of the 22th Annual Symposium on Computer* Architecture, pp. 24-36, junho, 1995.

[87] ZHOU Y; IFTODE L.; LI K. *Performance Evaluation of Two Home-Based Lazy Release Consistency Protocols for Shared Virtual Memory Systems*. Artigo de evento. Proceedings of the 2nd Symposium on Operating Systems Design and Implementation, outubro, 1996.

BOLETINS TÉCNICOS - TEXTOS PUBLICADOS

BT/PCS/9301 - Interligação de Processadores através de Chaves Ômicron - GERALDO LINO DE CAMPOS, DEMI GETSCHKO

BT/PCS/9302 - Implementação de Transparência em Sistema Distribuído - LUÍSA YUMIKO AKAO, JOÃO JOSÉ NETO

BT/PCS/9303 - Desenvolvimento de Sistemas Especificados em SDL - SIDNEI H. TANO, SELMA S. S. MELNIKOFF

BT/PCS/9304 - Um Modelo Formal para Sistemas Digitais à Nível de Transferência de Registradores - JOSÉ EDUARDO MOREIRA, WILSON VICENTE RUGGIERO

BT/PCS/9305 - Uma Ferramenta para o Desenvolvimento de Protótipos de Programas Concorrentes - JORGE KINOSHITA, JOÃO JOSÉ NETO

BT/PCS/9306 - Uma Ferramenta de Monitoração para um Núcleo de Resolução Distribuída de Problemas Orientado a Objetos - JAIME SIMÃO SICHMAN, ELERI CARDOSO

BT/PCS/9307 - Uma Análise das Técnicas Reversíveis de Compressão de Dados - MÁRIO CESAR GOMES SEGURA, EDIT GRASSIANI LINO DE CAMPOS

BT/PCS/9308 - Proposta de Rede Digital de Sistemas Integrados para Navio - CESAR DE ALVARENGA JACOBY, MOACYR MARTUCCI JR.

BT/PCS/9309 - Sistemas UNIX para Tempo Real - PAULO CESAR CORIGLIANO, JOÃO JOSÉ NETO

BT/PCS/9310 - Projeto de uma Unidade de Matching Store baseada em Memória Paginada para uma Máquina Fluxo de Dados Distribuído - EDUARDO MARQUES, CLAUDIO KIRNER

BT/PCS/9401 - Implementação de Arquiteturas Abertas: Uma Aplicação na Automação da Manufatura - JORGE LUIS RISCO BECERRA, MOACYR MARTUCCI JR.

BT/PCS/9402 - Modelamento Geométrico usando do Operadores Topológicos de Euler - GERALDO MACIEL DA FONSECA, MARIA ALICE GRIGAS VARELLA FERREIRA

BT/PCS/9403 - Segmentação de Imagens aplicada a Reconhecimento Automático de Alvos - LEONCIO CLARO DE BARROS NETO, ANTONIO MARCOS DE AGUIRRA MASSOLA

BT/PCS/9404 - Metodologia e Ambiente para Reutilização de Software Baseado em Composição - LEONARDO PUJATTI, MARIA ALICE GRIGAS VARELLA FERREIRA

BT/PCS/9405 - Desenvolvimento de uma Solução para a Supervisão e Integração de Células de Manufatura Discreta - JOSÉ BENEDITO DE ALMEIDA, JOSÉ SIDNEI COLOMBO MARTINI

BT/PCS/9406 - Método de Teste de Sincronização para Programas em ADA - EDUARDO T. MATSUDA, SELMA SHIN SHIMIZU MELNIKOFF

BT/PCS/9407 - Um Compilador Paralelizante com Detecção de Paralelismo na Linguagem Intermediária - HSUEH TSUNG HSIANG, LÍRIA MATSUMOTO SAITO

BT/PCS/9408 - Modelamento de Sistemas com Redes de Petri Interpretadas - CARLOS ALBERTO SANGIORGIO, WILSON V. RUGGIERO

BT/PCS/9501 - Sintese de Voz com Qualidade - EVANDRO BACCI GOUVÊA, GERALDO LINO DE CAMPOS

BT/PCS/9502 - Um Simulador de Arquiteturas de Computadores "A Computer Architecture Simulator" - CLAUDIO A. PRADO, WILSON V. RUGGIERO

BT/PCS/9503 - Simulador para Avaliação da Confiabilidade de Sistemas Redundantes com Reparo - ANDRÉA LUCIA BRAGA, FRANCISCO JOSÉ DE OLIVEIRA DIAS

BT/PCS/9504 - Projeto Conceitual e Projeto Básico do Nível de Coordenação de um Sistema Aberto de Automação, Utilizando Conceitos de Orientação a Objetos - NELSON TANOMARU, MOACYR MARTUCCI JUNIOR

BT/PCS/9505 - Uma Experiência no Gerenciamento da Produção de Software - RICARDO LUIS DE AZEVEDO DA ROCHA, JOÃO JOSÉ NETO

BT/PCS/9506 - MétodOO - Método de Desenvolvimento de Sistemas Orientado a Objetos: Uma Abordagem Integrada à Análise Estruturada e Redes de Petri - KECHI HIRAMA, SELMA SHIN SHIMIZU MELNIKOFF

BT/PCS/9601 - MOOPP: Uma Metodologia Orientada a Objetos para Desenvolvimento de Software para Processamento Paralelo - ELISA HATSUE MORIYA HUZITA, LÍRIA MATSUMOTO SATO

BT/PCS/9602 - Estudo do Espalhamento Brillouin Estimulado em Fibras Ópticas Monomodo - LUIS MEREGE SANCHES, CHARLES ARTUR SANTOS DE OLIVEIRA

BT/PCS/9603 - Programação Paralela com Variáveis Compartilhadas para Sistemas Distribuídos - LUCIANA BEZERRA ARANTES, LIRIA MATSUMOTO SATO

BT/PCS/9604 - Uma Metodologia de Projeto de Redes Locais - TEREZA CRISTINA MELO DE BRITO CARVALHO, WILSON VICENTE RUGGIERO

BT/PCS/9605 - Desenvolvimento de Sistema para Conversão de Textos em Fonemas no Idioma Português - DIMAS TREVIZAN CHBANE, GERALDO LINO DE CAMPOS

BT/PCS/9606 - Sincronização de Fluxos Multimídia em um Sistema de Videoconferência - EDUARDO S. C. TAKAHASHI, STEFANIA STIUBIENER

BT/PCS/9607 - A importância da Completeza na Especificação de Sistemas de Segurança - JOÃO BATISTA CAMARGO JÚNIOR, BENÍCIO JOSÉ DE SOUZA

BT/PCS/9608 - Uma Abordagem Paraconsistente Baseada em Lógica Evidencial para Tratar Exceções em Sistemas de Frames com Múltipla Herança - BRÁULIO COELHO ÁVILA, MÁRCIO RILLO

BT/PCS/9609 - Implementação de Engenharia Simultânea - MARCIO MOREIRA DA SILVA, MOACYR MARTUCCI JÚNIOR

BT/PCS/9610 - Statecharts Adaptativos - Um Exemplo de Aplicação do STAD - JORGE RADY DE ALMEIDA JUNIOR, JOÃO JOSÉ NETO

BT/PCS/9611 - Um Meta-Editor Dirigido por Sintaxe - MARGARETE KEIKO IWAI, JOÃO JOSÉ NETO

BT/PCS/9612 - Reutilização em Software Orientado a Objetos: Um Estudo Empírico para Analisar a Dificuldade de Localização e Entendimento de Classes - SELMA SHIN SHIMIZU MELNIKOFF, PEDRO ALEXANDRE DE OLIVEIRA GIOVANI

BT/PCS/9613 - Representação de Estruturas de Conhecimento em Sistemas de Banco de Dados - JUDITH PAVÓN MENDONZA, EDIT GRASSIANI LINO DE CAMPOS

BT/PCS/9701 - Uma Experiência na Construção de um Tradutor Inglês - Português - JORGE KINOSHITA, JOÃO JOSÉ NETO

BT/PCS/9702 - Combinando Análise de "Wavelet" e Análise Entrópica para Avaliar os Fenômenos de Difusão e Correlação - RUI CHUO HUEI CHIOU, MARIA ALICE G. V. FERREIRA

BT/PCS/9703 - Um Método para Desenvolvimento de Sistemas de Computacionais de Apoio a Projetos de Engenharia - JOSÉ EDUARDO ZINDEL DEBONI, JOSÉ SIDNEI COLOMBO MARTINI

BT/PCS/9704 - O Sistema de Posicionamento Global (GPS) e suas Aplicações - SÉRGIO MIRANDA PAZ, CARLOS EDUARDO CUGNASCA

BT/PCS/9705 - METAMBI-OO - Um Ambiente de Apoio ao Aprendizado da Técnica Orientada a Objetos - JOÃO UMBERTO FURQUIM DE SOUZA, SELMA S. S. MELNIKOFF

BT/PCS/9706 - Um Ambiente Interativo para Visualização do Comportamento Dinâmico de Algoritmos - IZAURA CRISTINA ARAÚJO, JOÃO JOSÉ NETO

BT/PCS/9707 - Metodologia Orientada a Objetos e sua Aplicação em Sistemas de CAD Baseado em "Features" - CARLOS CÉSAR TANAKA, MARIA ALICE GRIGAS VARELLA FERREIRA

BT/PCS/9708 - Um Tutor Inteligente para Análise Orientada a Objetos - MARIA EMÍLIA GOMES SOBRAL, MARIA ALICE GRIGAS VARELLA FERREIRA

BT/PCS/9709 - Metodologia para Seleção de Solução de Sistema de Aquisição de Dados para Aplicações de Pequeno Porte - MARCELO FINGUERMAN, JOSÉ SIDNEI COLOMBO MARTINI

BT/PCS/9801 - Conexões Virtuais em Redes ATM e Escalabilidade de Sistemas de Transmissão de Dados sem Conexão - WAGNER LUIZ ZUCCHI, WILSON VICENTE RUGGIERO

BT/PCS/9802 - Estudo Comparativo dos Sistemas da Qualidade - EDISON SPINA, MOACYR MARTUCCI JR.

BT/PCS/9803 – The VIBRA Multi-Agent Architecture: Integrating Purposive Vision With Deliberative and Reactive Planning – REINALDO A. C. BIANCHI , ANNA H. REALI C. RILLO, LELIANE N. BARROS

BT/PCS/9901 – Metodologia ODP para o Desenvolvimento de Sistemas Abertos de Automação – JORGE LUIS RISCO BECCERRA, MOACYR MARTUCCI JUNIOR

BT/PCS/9902 – Especificação de Um Modelo de Dados Bitemporal Orientado a Objetos – SOLANGE NICE ALVES DE SOUZA, EDIT GRASSIANI LINO DE CAMPOS

BT/PCS/9903 – Implementação Paralela Distribuída da Dissecação Cartesiana Aninhada – HILTON GARCIA FERNANDES, LIRIA MATSUMOTO SATO

BT/PCS/9904 – Metodologia para Especificação e Implementação de Solução de Gerenciamento – SERGIO CLEMENTE, TEREZA CRISTINA MELO DE BRITO CARVALHO

BT/PCS/9905 – Modelagem de Ferramenta Hipermídia Aberta para a Produção de Tutoriais Interativos – LEILA HYODO, ROMERO TORI

BT/PCS/9906 – Métodos de Aplicações da Lógica Paraconsistente Anotada de Anotação com Dois Valores-LPA2v com Construção de Algoritmo e Implementação de Circuitos Eletrônicos – JOÃO I. DA SILVA FILHO, JAIR MINORO ABE

BT/PCS/9907 – Modelo Nebuloso de Confiabilidade Baseado no Modelo de Markov – PAULO SÉRGIO CUGNASCA, MARCO TÚLIO CARVALHO DE ANDRADE

BT/PCS/9908 – Uma Análise Comparativa do Fluxo de Mensagens entre os Modelos da Rede Contractual (RC) e Colisões Baseada em Dependências (CBD) – MÁRCIA ITO, JAIME SIMÃO SICHMAN

BT/PCS/9909 – Otimização de Processo de Inserção Automática de Componentes Eletrônicos Empregando a Técnica de Times Assíncronos – CESAR SCARPINI RABAK, JAIME SIMÃO SICHMAN

BT/PCS/9910 – MIISA – Uma Metodologia para Integração da Informação em Sistemas Abertos – HILDA CARVALHO DE OLIVEIRA, SELMA S. S. MELNICOFF

BT/PCS/9911 – Metodologia para Utilização de Componentes de Software: um estudo de Caso – KAZUTOSI TAKATA, SELMA S. S. MELNIKOFF

BT/PCS/0001 – Método para Engenharia de Requisitos Norteado por Necessidades de Informação – ARISTIDES NOVELLI FILHO, MARIA ALICE GRIGAS VARELLA FERREIRA

BT/PCS/0002 – Um Método de Escolha Automática de Soluções Usando Tecnologia Adaptativa – RICARDO LUIS DE AZEVEDO DA ROCHA, JOÃO JOSÉ NETO

BT/PCS/0101 – Gerenciamento Hierárquico de Falhas – JAMIL KALIL NAUFAL JR., JOÃO BATISTA CAMARGO JR.

BT/PCS/0102 – Um Método para a Construção de Analisadores Morfológicos, Aplicado à Língua Portuguesa, Baseado em Autômatos Adaptativos – CARLOS EDUARDO DANTAS DE MENEZES, JOÃO JOSÉ NETO

BT/PCS/0103 – Educação pela Web: Metodologia e Ferramenta de Elaboração de Cursos com Navegação Dinâmica – LUISA ALEYDA GARCIA GONZÁLEZ, WILSON VICENTE RUGGIERO

BT/PCS/0104 – O Desenvolvimento de Sistemas Baseados em Componentes a Partir da Visão de Objetos – RENATA EVANGELISTA ROMARIZ RECCO, JOÃO BATISTA CAMARGO JÚNIOR

BT/PCS/0105 – Introdução às Gramáticas Adaptativas – MARGARETE KEIKO IWAI, JOÃO JOSÉ NETO

BT/PCS/0106 – Automação dos Processos de Controle de Qualidade da Água e Esgoto em Laboratório de Controle Sanitário – JOSÉ BENEDITO DE ALMEIDA, JOSÉ SIDNEI COLOMBO MARTINI

BT/PCS/01/07 – Um Mecanismo para Distribuição Segura de Vídeo MPEG – CÍNTIA BORGES MARGI, GRAÇA BESSAN, WILSON VICENTE RUGGIERO

BT/PCS/0108 – A Dependence-Based Model for Social Reasoning in Multi-Agent Systems – JAIME SIMÃO SICHMAN

BT/PCS/0109 – Ambiente Multilinguagem de Programação – Aspectos do Projeto e Implementação – APARECIDO VALDEMIR DE FREITAS, JOÃO JOSÉ NETO

BT/PCS/0110 – LETAC: Técnica para Análise de Tarefas e Especificação de Fluxo de Trabalho Cooperativo – MARCOS ROBERTO GREINER, LUCIA VILELA LEITE FILGUEIRAS

BT/PCS/0111 – Modelagem ODP para o Planejamento de Sistemas de Potência – ANIRIO SALLES FILHO, JOSÉ SIDNEI COLOMBO MARTINI

BT/PCS/0112 – Técnica para Ajuste dos Coeficientes de Quantização do Padrão MPEG em Tempo Real – REGINA M. SILVEIRA, WILSON V. RUGGIERO

BT/PCS/0113 – Segmentação de Imagens por Classificação de Cores: Uma Abordagem Neural – ALEXANDRE S. SIMÕES, ANNA REALI COSTA