EFEITO DO MULTICASTING NA COERÊNCIA DE CACHE COM DIRETÓRIOS "FULL MAP" NUM SISTEMA MULTIPROCESSADOR COM REDE CROSSBAR

Edward D.M. Ordoñez, **Martha X.T.Delgado, Celso A.S.Santos, Sergio T. Kofuji.** [1]

Laboratório de Sistemas Integráveis
Departamento de Engenharia Eletrônica
Escola Politécnica da Universidade de São Paulo

RESUMO

No presente trabalho especifica-se e analisa-se um protocolo de coerência de cache baseado nos esquemas de diretório distribuído "full-map" para o "cluster" de um sistema multiprocessador com rede de interconexão crossbar com propriedade de "multicasting". A análise de desempenho considerada é baseada nas transmissões feitas através da rede (tráfego da rede), para três protocolos considerados (um protocolo "snoopy", esquema diretório com rede crossbar clássica, e a nossa proposta: esquema diretório com crossbar "multicasting") e estabelecem-se diferenças no seu desempenho.

ABSTRACT

This paper analyses a cache coherence protocol based on the schemes of the directory full-map distributed for the cluster of a multiprocessor system with crossbar network interconection, with multicasting property. The performance analyses considered is based on the transmitions made through the network, the traffic through it, for three protocols considered (snooopy protocol, directory schemes case with classic crossbar network and our proposal: directory schemes with multicasting crossbar). Differences on their performance are shown.

[1] Pesquisadores do Laboratório de Sistema Integráveis (LSI/EPUSP)
E-mail: (edmoreno, mxtd, saibel, kofuji)@lsi.usp.br

1. INTRODUÇÃO.

A consideração de memórias caches privadas associadas a cada processador reduz os problemas de contenção de memória e de comunicação, mas geram o problema de que os caches podem potencialmente ter várias cópias de blocos compartilhados em um ou mais caches no mesmo tempo, originando assim inconsistência dos dados compartilhados entre os caches e entre os caches e a memória. É preciso ter uma visão coerente dos sistema de memória, este é o problema de coerência de cache. Existem muitos estudos sobre coerência de cache e as soluções podem ser baseadas em hardware ou em software [1].

As soluções baseadas em software são atraentes porque requerem um suporte mínimo de hardware, podem escalar até limites impostos pela rede de interconexão e o overhead de deteção de dados é feita no tempo de compilação (e não no tempo de execução). A coerência com software pode ser feita pela análise do programa, os blocos de memória são rotulados como armazenados em cache ou não, esta operação pode ser feita por um pré-processador ou pelo compilador. Estas soluções tem desvantagens tais como: nos esquemas simples a especificação das estruturas compartilhadas e a sua sincronização são feitas pelo programador ou compilador, requerem que o compartilhamento seja através da memória, e os mecanismos que forçam a coerência são gerados pelo compilador estaticamente. ***As soluções baseadas em hardware*** são muito utilizadas e liberam o programador e/ou compilador da responsabilidade de especificar as operações de consistência para estruturas compartilhadas; estas propostas acarretam um custo na complexidade do hardware (do cache, do controlador da memória). Estas propostas podem ser classificadas como protocolos "snoopy" e esquemas baseados em diretórios [1,2].

Os protocolos "snoopy" geralmente são utilizados em arquiteturas baseadas em barramento. Neste caso, as modificações de cada bloco de cache são transmitidas a todos os caches e cada um deles monitora o barramento para estabelecer as ações de consistência (invalidações e/ou atualizações). A implementação destes protocolos é simples mas tem a grande desvantagem de utilizarem o barramento para as transmissões dos comandos de consistência, e isto reduz a largura de banda da rede de interconexão, apresentando assim baixa escalabilidade.

Os protocolos baseados em esquemas diretório armazenam a informação sobre onde residem as cópias dos blocos e os comandos de consistência podem ser transmitidos só aqueles caches que possuem cópias do bloco de interesse. Estas soluções tipicamente armazenam a informação na memória, com uma entrada por cada bloco contendo o estado possível de cada bloco e um vetor com um bit por cada cache indicando quais deles tem uma cópia do bloco em questão. Estes esquemas são mais escaláveis que os protocolos "snoopy" mas têm o problema de precisar de mais memória e que a informação dos estados é armazenada na memória (e não nos caches).

Neste documento, propõe-se um mecanismo de coerência de caches baseado nos esquemas diretório (se utiliza um diretório "full-map" -apresenta um vetor que tem um bit por cada cache para indicar o(s) cache(s) que possuem cópia do bloco - assim só os caches com cópias podem ser afetados por um acesso a um bloco de memória e só aqueles caches são informados dos acessos) com a diferença que a informação dos estados (e do diretório em geral) é armazenada e distribuída nos caches.

A idéia nova é a utilização de um **crossbar** (rede clásica de interconexão) **com mecanismos especiais que facilitem a coerência de cache.** Com os avanços tecnológicos que permitem o projeto de sistemas VLSI em Arseneto de Gálio GaAs e de novas técnicas que melhoram o desempenho dos atuais sistemas, tem sido possível construir sistemas crossbar de melhor desempenho e com custos comparáveis a outras redes de interconexão. Por exemplo, já foi projetado um crossbar de 256x256 eficiente a uma velocidade de transmissão até 250 Mbps por linha e que pode ser implementado em um chip de 1cm2 com tecnologia CMOS VLSI [3] e implementação de técnicas de arbitração em crossbar que melhoram o "throughput" da rede em mais de um 40% [4]. É importante ressaltar que a utilização de memórias caches minimizam o efeito de latência na rede de interconexão, neste caso o crossbar.

Na seção seguinte (seção 2) são apresentadas as motivações deste trabalho. Na seção 3 apresenta-se a organização do sistema e a descrição de seu protocolo de coerência, dando ênfase na utilização do crossbar com apoio a coerência de cache. Na seção 4 são expostos os resultados do trabalho e na seção 5 são discutidas algumas conclusões e propostas para desenvolver futuros trabalhos.

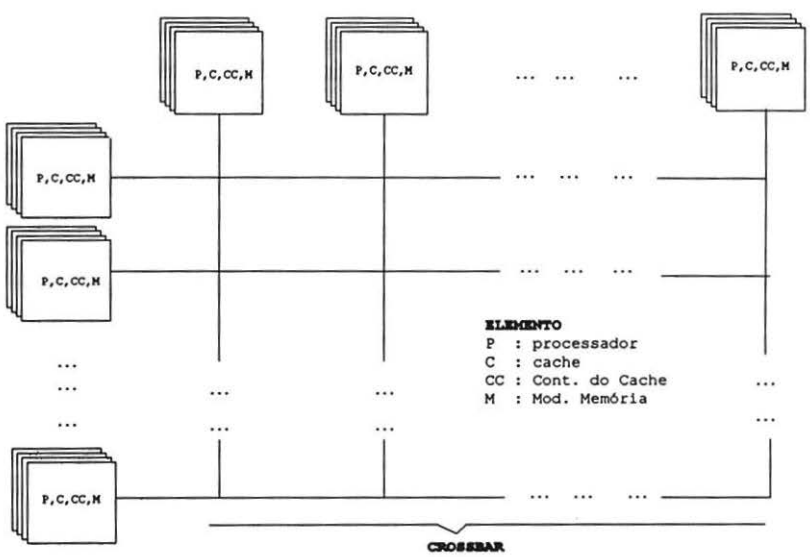

Figura 1. Sistema Multiprocessador baseado em clusters interconectados com crossbar.

2. MOTIVAÇÃO.

Nos últimos anos tem-se verificado um grande progresso em computação de alto desempenho. Por exemplo, existem várias máquinas com desempenho da ordem de 10 Gflops (Cray YMP C90, Thinking Machines CM-5, NEC SX-3, HItachi, Intel Paragon). Não obstante, precisa-se de sistemas maciçamente paralelos com maior velocidade (do ordem dos Teraflops) com o objetivo de pesquisar problemas científicos e de engenharia conhecidos como "Grand

Challenges". No momento, companhias de supercomputadores têm anunciado sistemas maciçamente paralelos baseados em multiprocessadores comerciais (Convex com o RISC HP-PA, Cray com o DEC Alpha, Thinking Machines com o SPARC). Por exemplo, a "Cray Computer" proclamou (finais do 93-começo do 94) uma máquina de memória compartilhada maciçamente paralela com desempenho próximo aos Teraflops (o desempenho máximo desta máquina seria de 200-400 Gflops)[5].

A tendência atual é projetar sistemas com maior potência computacional a custos relativamente baixos. Estes sistemas são baseados em sistemas multiprocessadores formados por "clusters" ligados com redes de interconexão com tempo de latência aceitável, já que o tráfego entre os processadores e a memória não-local é muito intenso e a rede de interconexão deve ser capaz de manobrar este tráfego com uma latência razoável. Por exemplo, existem sistemas cujos nós de processamento ("clusters") são conectados com duas redes Mesh (caso do protótipo DASH [6,7] e outros sistemas que utilizam anéis de fibras óticas (caso do SCI "Scalable Coherent Interface" [8]). Estes sistemas DASH e SCI implementam a coerência de cache com propostas baseadas em esquemas diretório.

O nosso objetivo é aproveitar os novos avanços obtidos com redes clásicas (como o crossbar) [3,4] e propor um mecanismo de coerência de cache baseado em esquemas diretório em um sistema relativamente grande. Por exemplo, implementar um sistema baseado em "clusters" e conectá-los com um crossbar, estabelecendo um protocolo de coerência de cache baseado em esquemas diretório. Com o novo crossbar de 256x256 e "clusters" de 4 processadores poderia ser obtido um sistema com até 1024 processadores.

A figura 1 mostra a organização do sistema baseado em "clusters" conectados através de um crossbar. A sua coerência de cache pode ser mantida com um protocolo baseado em esquemas de diretório distribuído. No "cluster", os processadores podem estar conectados através de um barramento (tem mostrado eficiência para poucos processadores) e a sua coerência pode ser mantida com os bastante conhecidos protocolos "snoopy" [1,2,9].

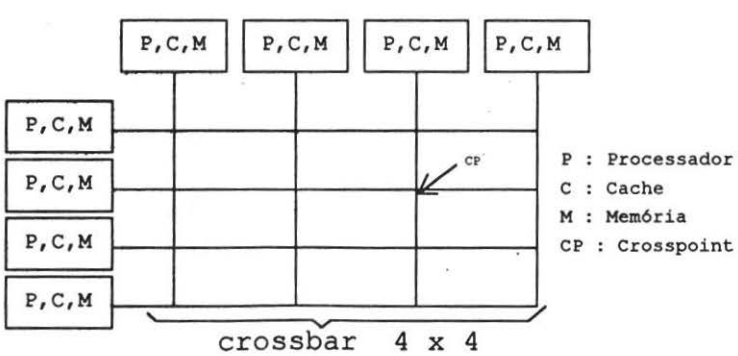

Figura 2. Arquitetura do Multiprocessador.

Outra possibilidade é conectar os processadores, no "cluster", através de um crossbar (aproveitar a sua maior escalabilidade para assim formar "clusters" maiores), como se observa na figura 2.

Como os esquemas baseados em diretórios distribuídos não são muito conhecidos e estudados (em relação aos protocolos "snoopy") e com a idéia de utilizar o crossbar, é implementado e estudado um protocolo para esta arquitetura de "clusters" formados por 4 processadores. Este protocolo de coerência de cache é analisado na seção seguinte.

3. ORGANIZAÇÃO DO SISTEMA E PROTOCOLO DE COERÊNCIA DE CACHE.

3.1 Organização do Sistema.

Na figura 2 é mostrada a arquitetura adotada. O sistema consiste de quatro elementos de processamento conectados por uma rede de interconexão crossbar. Cada elemento de processamento é formado por um processador, um cache, um controlador de cache e um módulo de memória. Ter o cache e a memória de um nó ligados diretamente permite que a comunicação entre estes dois elementos seja rápida e não ocorra pela rede de interconexão (assim consegue-se reduzir o tráfego pela rede). A rede de interconexão somente é utilizada nos casos de transferência cache-cache ou transferência mamória-cache de nós diferentes. As informações transmitidas pela rede restrimgem-se a dados compartilhados e estados do controle estabelecido pelo protocolo de coerência.

A rede de interconexão crossbar considerada é uma rede de chaveamento de pacotes que permite receber pacotes de dados existentes em seus canais de entrada e roteá-los apropriadamente a seus canais de saída. A largura de banda das entradas é tipicamente igual à largura de banda das saídas. Neste estudo consideram-se as possibilidades de construir um crossbar com a propriedade de ter um maior número de conexões simultaneamente. Com este fim devemos levar em conta a existência das técnicas de arbitragem discutidas e estabelecidas por Tamir e Hsin-Chou que permitem melhorar o desempenho de pacotes de processamento não-FIFO (especificamente, dados de uma determinada entrada destinados a diferentes saídas podem ser transmitidos através da chave em qualquer ordem). A arbitragem no crossbar permite que um número máximo de pacotes de dados sejam transmitidos simultaneamente. Por exemplo, o uso de buffers DAMQ "Dinamically Allocated MultiQueue" permite que um número maior de pontos de cruzamento "crosspoint" do crossbar sejam conectados, e isto produz um maior "throughput" e uma menor latência da rede [4].

3.2 Protocolo de Coerência de Cache.

O protocolo de coerência de cache utilizado é um protocolo de hardware baseado em esquemas de diretório distribuído. A informação dos estados de coerência é localizada nos caches privados. Ele implementa um esquema de coerência baseado em invalidações, utilizando uma política WRITE-BACK.

3.2.1 Estados do Cache.

O protocolo possui os seguintes estados de um bloco de cache: INVÁLIDO, PRIVADO, CONSISTENTE e COMPARTILHADO. Na figura 3, mostra-se o estado, seu significado e o conteúdo do campo de estados do i-ésimo cache. Cada cache deve ter uma tabela contendo em cada entrada: *o campo de dados* (mantém a cópia do bloco) , *o campo de rótulo* ("tag" mantém a identificação do bloco que ocupa a entrada cache) *e o campo de estados* (mantém a informação que determina as ações que devem ser tomadas pelo protocolo de coerência de cache). Cada bloco nos módulos de memória contém também uma entrada com um bit de validade e bits que indicam o estado possível do bloco.

O campo de estados deve ser constituído pelos seguintes bits:
- um bit de validade (V): indica se a cópia do bloco é válida (1) ou não (0).
- um bit de modificação (M): indica se é necessário uma operação write-back (WB). operação WB (1), não WB (0).
- um vetor de ponteiros (Ci): um ponteiro or cada cache para indicar os caches que possuem cópia do bloco. Ci=1 esta no cache i.
- um vetor de estados (Ei): indica o possível estado do bloco. Precisa de dois bits, máximo quatro estados.

ESTADO	SIGNIFICADO	CAMPO DE ESTADOS
INVÁLIDO	a cópia é inconsistente	Vi=0, Mi=0, E1=0, E0=0
PRIVADO	a cópia existe só num cache e a memória é inconsistente	Vi=1, Mi=1, E1=0, E0=1
CONSISTENTE	a cópia existe no cache e a memória é consistente.	Vi=1, Mi=0, E1=1, E0=0
COMPARTILHADO	a cópia existe em vários caches	Vi=1, Mi=1, E1=1, E0=0

Figura 3. Informação dos estados e seu significado.

Na figura 4, mostra-se a informação armazenada nos caches no caso de quatro processadores com seus caches privados. Duas cópias de um bloco com identificação R existem nos caches 1 e 3; os bits M1 e M3 com valor 1 indicam que a cópia foi modificada (é consistente com a cópia da memória) então deve-se fazer uma operação copy-back. Os ponteiros de cache Ci=1 (i=1,3) indicam que os caches C1 e C3 possuem cópia do bloco (os outros Ci=0). A cópia que eles possuem é válida Vi=1 (i=1,3). O estado deste bloco é compartilhado (indicado por E1 e Eo com valor 1). O cache 0 possui uma cópia inválida (Vo=0) e o seu estado é inválido Eo=E1=0 e o cache 2 não possue cópia do bloco (a identificação deste bloco é T).

3.2.2 Especificações do Protocolo.

As ações tomadas pelo protocolo dependem das operações realizadas pelos processadores (leitura ou escrita). Portanto, o comportamento do protocolo é descrito com base nas ações apropriadas sobre os quatro resultados de uma operação de leitura ou de escrita, denominadas: "read hit", "read miss", "write hit", "write miss". Entende-se como "hit" que a cópia é válida e existe no cache e como "miss" que a cópia é inválida ou não existe.

1. READ HIT:
Um processador Pi faz o pedido de leitura de um bloco localizado em um endereço determinado, que pode ser indicado por *(ende)*. A cópia do bloco existe no seu cache correspondente (Ci) e é válida, indicada pelo bit Vi=1. A operação é feita localmente e não existe nenhuma modificação.

2. READ MISS:
Um processador Pi faz um pedido de leitura de um bloco com endereço *ende*. A cópia não existe no seu cache correspondente. Podem ocorrer as seguintes possibilidades:

- A cópia existe na memória. O módulo de memória deve fornecer o dado ao cache, comando *get(CCi,ende)*. Esta transferência é seguida por uma mudança de estados *Novoestado(ende,PCi, Vi,Mi,Eij)*. Neste caso: PCi=000Ci (Ci o cache que contem a cópia), Vi=1, Mi=0 (indica que um próximo acesso a este bloco não precisa de uma operação de copy-back -memória contém a cópia do bloco), Eij=10 (estado CONSISTENTE).

- A cópia existe em cache. O controlador de cache solicita a cópia ao(s) cache(s) que possui(em) a cópia (estes são identificados pelos ponteiros de cache PCi). No caso de existir em vários caches, só um deles deve fornecer o dado, comando *put(CCi, ende)*. Enquanto a cópia é transferida pela rede de interconexão, se faz um copy-back nos caches proprietários e uma mudança de estados assim: *Novoestado(ende,PCi,1,0,1,1)*. O novo estado é COMPARTILHADO. Quando o dado é recebido pelo controlador do cache solicitante, se faz a escrita em memória e se transmite ao cache solicitante, comando *get(CCi, ende)*. Esta última transferência é seguida por uma mudança na entrada cache, *Novoestado(ende,PCI,1,0,1,1)*. O novo estado é COMPARTILHADO.

	TAG	C3	C2	C1	C0	V	M	E1	E0
cache 0	R	-	-	-	-	0	-	0	0
cache 1	R	1	0	1	0	1	1	1	1
cache 2	T	-							-
cache 3	R	1	0	1	0	1	1	1	1

Figura 4. Informação dos estados distribuídos nos caches.

3. WRITE HIT:

Um processador faz uma escrita em um bloco no endereço *ende*. O bloco se encontra no cache correspondente. Dependendo do estado do bloco, ocorrem as seguintes ações:

- A cópia está no estado INVÁLIDO. Se faz a escrita localmente, comando *store(ende,index)*. Armazena a cópia no endereço *ende*, o valor *index* dentro do bloco. O comando *Novoestado (ende, PCi,1,1,0,1)* atualiza o campo de estados. O novo estado é PRIVADO.

- A cópia está no estado PRIVADO (existe só num cache). Só o PCi respectivo é ativo. Antes de fazer a escrita local, se verifica o bit de consistência. Se Mi=0 se faz a escrita diretamente, se Mi=1 deve-se fazer primeiramente um copy-back e depois a escrita é feita localmente. Se faz uma atualização de estados, comando *Novoestado(ende, PCi,1,1,0,1)*. O novo estado é PRIVADO.

- A cópia está no estado COMPARTILHADO. Vários PCi estão ativos. A cópia é armazenada no cache corespondente, comando *store(ende, index)*. O controlador do cache transmite um sinal de invalidação a todos os outros caches, indicados pelos PCi -invalidando seus valores. O comando *Novoestado(ende,PCi,1,1,0,1)* atualiza o campo de estados. O estado é PRIVADO.

As outras cópias podem ser invalidadas em todos os outros caches simultaneamente, operação permitida pelo crossbar considerado.

4. WRITE MISS:

Podem ocorrer as seguintes possibilidades :

- A cópia é armazenada no módulo de memória correspondente. O estado na memória é válido e não existe transferência pela rede de interconexão.
- A cópia não existe em cache. O bloco é levado à memória correspondente. Não existe transferência pela rede, a escrita se faz localmente no módulo de memória. O comando Novoestado(ende,PCi,1,0,0,1) troca o estado na memória. O novo estado é PRIVADO.
- A cópia existe em vários caches. O bloco é transmitido à memória com uma mudança de estados na memória, o novo estado é PRIVADO. Simultaneamente se transmite uma invalidação a todos os caches que possuem cópia. Existe transferência pela rede de interconexão.

5. SUBSTITUIÇÃO:

O protocolo começa com uma instrução *load(ende, index)* ou *store(ende,index)* de um processador. Dependendo do estado do bit de validade V*i* e do bit de consistência M*i*, tem-se os seguintes casos:

- bit Vi=0; não é preciso fazer uma substituição.
- bit Vi=1 e Mi=0; o bloco não precisa ser substituído.
- bit Vi=1 e Mi=1; o bloco no cache é transmitido à memória, comando *put(copy, anende)*. (put (cópia, anende)) transmite a cópia no endereço antigo *anende* que é substituído.

3.3 Operações de Transferência.

O crossbar utiliza uma rede de chaveamento de pacotes que permite realizar operações de "multicasting", as quais ajudam a melhorar o desempenho do sistema e permitem um comportamento semelhante as de uma rede de interconexão com bom desempenho no sistema de coerência de cache. Por exemplo, na figura 5 mostram-se as vantagens desta rede comparada com as redes clássicas.

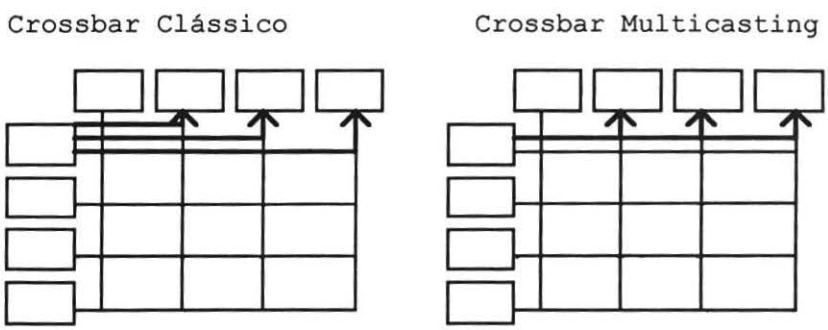

Figura 5. Operações de multicasting no crossbar.

4. AVALIAÇÃO DE DESEMPENHO.

O objetivo é comparar o desempenho de nossa proposta com um protocolo baseado em esquemas diretório mas com uma rede crossbar clássica e com um protocolo "snoopy"com uma rede baseada em barramento (o protocolo "write once" [12]), com o fim de estabelecer as diferenças entre os protocolos snoopy e os baseados em esquemas diretório e as possíveis vantagens no uso de um crossbar "multicasting". Nós comparamos o desempenho tendo como base o **custo de comunicação** e verificamos que utilizando esta medida, a nossa proposta apresentava um menor custo devido ao efeito de "multicasting", apesar da diferença não ser muito significativa. Então fizemos uma análise com base no tráfego na rede de interconexão gerado pelos comandos de consistência do protocolo estabelecido, já que as transmissões aumentam a probabilidade de conflitos na rede e isto afeta o desempenho do sistema.

4.1 Análise.

As referências de memória no sistema podem ser a blocos no estado inválido, privado, consistente ou compartilhado. Na análise utilizam-se os seguintes valores: **W** é a probabilidade de que uma referência a um bloco compartilhado (BC) seja de escrita, **H** é a taxa de hits ("hit ratio") de BCs estarem no cache, **P** é a probabilidade de que a próxima referência seja a um BC, **N** é o número de caches no sistema (igual ao número de processadores), **P(ECOM)** é a probabilidade de um bloco estar no estado compartilhado, **P(EPR)** é a

probabilidade de um bloco estar no estado privado, **P(ECO)** é a probabilidade de um bloco estar no estado consistente. As transmissões pela rede ocorrem nos seguintes casos:

1. READ MISS:

Só é feita uma transmissão se o bloco está no estado privado e pertence a um cache de outro elemento de processamento. No caso do protocolo "write once" é necessário que (n-1) comandos sejam enviados a (n-1) caches, com o fim de detectar o cache que possui a cópia do bloco de interesse. *Tmem* é o tráfego gerado pela procura do bloco na memória e *Twb* é o tráfego gerado por uma operação de copy-back. No caso do protocolo diretório, Tem=Twb=0 já que não acontece pela rede, e neste caso só se envia um comando ao cache que possue a cópia (se conhece o cache, o qual é indicado pelos ponteiros de cache do protocolo proposto). Então, o tráfego para os três casos é:

- caso "write once": Trm(wo)= (N-1)*P*(1-H)*(1-W)*P(EPR)+Tmem+Twb
- caso crossbar clássico: Trm(cc)= P*(1-H)*(1-W)*P(EPR)
- caso crossbar "multicasting": Trm(cm)=Trm(cc)

2. WRITE HIT:

As transmissões são feitas somente se o bloco é compartilhado. No pior caso, é necessário enviar comandos a (n-1) caches. É preciso utilizar uma probabilidade condicional devido a que um bloco pode estar em pelo menos um cache.

-caso"write once": Twh(wo)=(n-1)*P*W*H*P(ECOM/(EPRI+ECOM)+ECO))+Tmem+Twb
- caso crossbar clássico: Twh(cc)= (N-1)*P*W*H*P(ECOM/(EPRI+ECOM+ECO))
- caso crossbar multicasting: Twh(cm)= P*W*H*P(ECOM/(EPRI+ECOM=ECO))

(no caso do multicasting aproveita-se que a transmissão se faz simultaneamente).

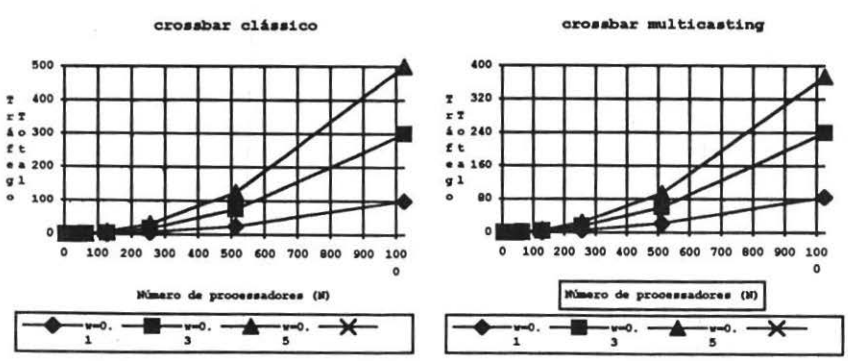

Figura 6. Comportamento do tráfego com variações de W.

3. WRITE MISS:

Se faz uma transmissão se o estado é diferente de inválido. Existem dois termos: um é para o tráfego caso o estado do bloco seja privado ou consistente e o outro termo para o caso compartilhado.

-Twm(wo)=(N-1)*P*W*(1-H)*(P(EPRI)+
P(ECO))+(N-1)*P*W*(1-H)*P(ECOM)+Tmem+Twb
- Twm(cc)= P*W*(1-H)*(P(EPRI)+P(ECO))+(N-1)*P*W*(1-H)*P(ECOM)

- Twm(cm)= P*W*(1-H)*(P(EPRI)+P(ECO))+P*W*(1-H)*P(ECOM)

Cada cache no sistema gera estes comandos, então o tráfego total observado por um cache devido a todos os (N-1) caches restantes é: **Ttotal (T)= (N-1)*(Trm+Twh+Twm).**

4.2 Resultados.

Como o tráfego é muito dependente do grau de compartilhamento, espera-se obter um bom desempenho (menor tráfego) no caso em que o compartilhamento é baixo. Assumem-se os seguintes valores [11]:
- Compartilhamento baixo: P=0.01, H=0.95, P(ECOM)=0.01, P(EPRI)=0.06 e P(ECO)=0.03
- Compartilhamento alto: P=0.10, H=0.80, P(ECOM)=0.10, P(EPRI)=0.35 e P(ECO)=0.35
(H é menor devido ao compartilhamento alto, P=0.10).

As figuras 6 e 7 mostram os resultados obtidos para diferentes valores de W (probabilidade de uma referência a um bloco compartilhado seja de escrita): W=0.1,0.3 e 0.5. e diferentes valores de N (número de processadores). Nestas figuras observa-se o comportamento do tráfego total (T) com o número de processadores (N).

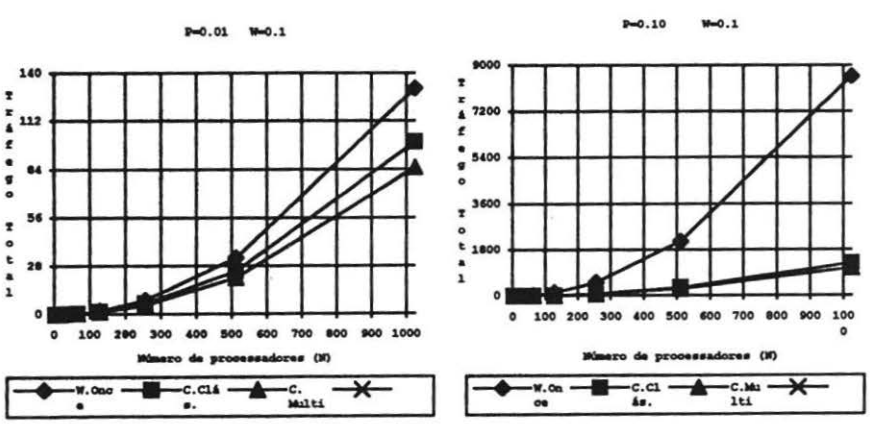

Figura 7. Comportamento do tráfego para as três propostas.

A figura 6 apresenta a dependência do tráfego para diferentes probabilidades W, para a proposta com crossbar clássico e crossbar multicasting no caso de compartilhamento baixo P=0.01. Na figura se observa que com o aumento de W se tem maior tráfego e portanto menor desempenho.

A figura 7 ilustra o comportamento do tráfego das três propostas de interesse. Nela observa-se que o caso do protocolo baseado em barramento possui um maior tráfego pela rede, o que confirma o fato que esta não é eficiente para muitos processadores. O tráfego pela rede crossbar clássica é menor que o anterior, devido a utilização do diretório que permite conhecer que caches possuem cópia de um determinado bloco, transmitindo comandos só para este(s).
A rede crossbar multicasting apresenta melhor desempenho que a rede clássica, o que torna possível que um número maior de processadores sejam conectados. É recomendável utilizar o crossbar multicasting nos casos em que o número de processadores seja superior a 512. Para um N menor a diferença no desempenho não é siginificativa e pode-se utilizar um crossbar clássico.

5. CONCLUSÕES.

Neste documento foi apresentado um protocolo de coerência de cache baseado em hardware e em esquemas de diretório distribuído, esquema "full-map" para um sistema multiprocessador cuja rede de interconexão é um crossbar com propriedade de "multicasting". A análise de desempenho utilizada foi feita com base nas transmissões necessárias pela rede de interconexão, fornecendo uma estimativa do tráfego na rede. Com base nesta análise, pode-se concluir que a proposta de utilizar um crossbar com propriedade de "multicasting" é razoável nos casos em que o número de processadores é maior que 512.

O método usado para avaliar o desempenho só fornece o custo de transmissão pela rede e não corresponde a uma medida de como o desempenho total do sistema é afetado. **Por este motivo, esta-se construído o simulador do sistema de memória da arquitetura em questão para a utilização do Tango-Lite, que é um ambiente de simulação projetado pela Universidade de Stanford** [12,13]. Com este simulador pode-se fazer uma análise mais precisa de desempenho, a fim de obter melhores estimativas do comportamento do sistema. Adicionalmente, deseja-se simular o sistema com variações no protocolo. Por exemplo, utilizar uma política "*write through*" e um esquema baseado em atualizações com o fim de estabelecer diferenças no desempenho do sistema.

Os protocolos de coerência de cache baseados em esquemas de diretório distribuído permitem obter maior escalabilidade em sistemas multiprocessadores com caches coerentes. O custo desta escalabilidade é a complexidade adicional desses esquemas comparados com esquemas "*snoopy*" existentes, baseados em barramento. Por este motivo, continuaremos estudando e aperfeiçoando o problema de consistência de cache com esquemas diretório. Em trabalhos futuros serão projetados sistemas multiprocessadores baseados em "clusters", e com coerência de cache baseada nestes esquemas (em diretórios distribuídos).

6. REFERÊNCIAS.

[1] EGGERS, S.J. Simulation Analysis of Data Sharing in Shared Memory Multiprocessors. **PhD Thesis**, Univeristy of California at Berkeley, Feb. 1989.

[2] STENSTRÖM, P. A survey of Cache Coherence Schemes for Multiprocessors. **IEEE Computer**, v. 23, n. 6, p. 12-24, Jun. 1990.

[3] CHO, K.; ADAMS,W.S. VLSI Implementation of a 256x256 Crossbar Interconnection Network. In: International Parallel Processing Symposium, IEEE. **Proceedings** 6th., p. 289-93, Mar. 1992.

[4] TAMIR, Y.; CHI, H.C. Symmetric Crossbar Arbiters for VLSI Communications Switches. **IEEE Transactions on Parallel and Distribuited Systems**, v. 4, n. 1, p. 13-27, Jan. 1993.

[5] MOREIRA, J.E. Proposal for a High Performance Scalable Multiprocessor. University of Illinois at Urbana-Champaign, Mar. 1993.

[6] LENOSKI, D.; LAUDON, J.; JOE, T.; NAKAHIRA, D.; STEVENS, L.; GUPTA, A.; HENNESSY, J. The DASH Prototype: Logic Overhead and Performance. **IEEE Transactions on Parallel and Distributed Systems**, v. 4, n. 1, p. 41-61, Jan. 1993.

[7] D.LENOSKI, D.; LAUDON, J.; GHARACHORLOO, K.; WEBER, W; GUPTA, A.; HENNESY, J.; HOROWITZ, M.; LAM, M.S. The Standford DASH Multiprocessor. **IEEE Computer**, , p. 63-79, Mar. 1992.

[8] P1596 Working Group, SCI (Scalable Coherence Interface) - Cache Coherence Overview. **Technical Report Revision 0.33**, IEEE Computer Society, Nov. 1989.

[9] PAPAMARCOS, M.S;.PATEL, J.H. A Low-Overhead Coherence Solution for Multiprocessors with Private Cache Memories. In: **Proceedings** Ann. Symp. on Computer Architecture, IEEE, 11th, p. 348-54, 1984

[10] ARCHIBALD, J; BAER, J.L. An Economical Solution to the Cache Coherence Problem. In: Ann. Symp. on Computer Architecture, IEEE, 11th, **Proceedings**, p. 355-62, 1984.

[11] GOODMAN, J.R. Using Cache Memory to Reduce Processor Memory Traffic. In: Ann. symp. on Computer Architecture, IEEE, **Proceedings** of 10th, p. 124-31, 1983.

[12] HERROD, S.A. Tango-Lite: A Multiprocessor Simulation Environment -Introduction and User's Guide. **Technical Report**, Stanford University, Sep. 1993.

[13] GOLDSMCHMIDT, S.R. Simulation of Multiprocessors: Accuracy and Performance. **PhD. Thesis**, DEE, University of Stanford, Jun. 1993.