CARACTERÍSTICA DE COMPARTILHAMENTO DOS PROGRAMAS CHOLESKY E MP3D

Sergio Takeo Kofuji, Martha Ximena Torres Delgado,
Edward David Moreno Ordoñez, João Antonio Zuffo

Laboratório de Sistemas Integráveis, Escola Politécnica da Universidade de São Paulo
Av. Prof. Luciano Gualberto, trav. 3, 158. CEP 05508-900 - São Paulo, SP, Brasil
tel: (011) 818-5667 fax: (011) 211-4574. (kofuji,martha, edmoreno,jazuffo)@lsi.usp.br

RESUMO

O **SPLASH** é um conjunto de programas aplicativos paralelos coletado pela Universidade de Stanford para uso em projeto e avaliação de sistemas multiprocessadores paralelos, distribuído livremente para fins acadêmicos. Utilizando simuladores baseados no **MINT**, será feita uma avaliação do comportamento de algumas aplicações do **SPLASH**. Três características serão estudadas, procurando estender os trabalhos anteriores da área e fornecer subsídios para a escolha de parâmetros arquiteturais do multiprocessador **SPADE**: ***i***) o custo da sincronização; ***ii***) a hierarquia de conjuntos de trabalho; e ***iii***) o número de cópias invalidadas em cada operação de escrita em caches que seguem a política de invalidação em escrita.

ABSTRACT

The Stanford SPLASH is a parallel suite to evaluate medium scale shared memory multiprocessing architectures. This work evaluates the sharing characteristics of some programs of the SPLASH suite, using the Rochester MINT simulator. In order to obtain information to decide architectural parameters of the SPADE, a large scale multiprocessor system in development at LSI-USP, three characteristics are studied: synchronization costs, sharing level, and working sets hierarchies.

1 - INTRODUÇÃO

O processamento paralelo é uma abordagem que tem se mostrado eficaz para a obtenção de alto desempenho. Há muitos anos o processamento paralelo em pequena escala tem sido explorado com sucesso em várias classes de sistemas de computação, desde estações de trabalho a computadores de grande porte e supercomputadores paralelo-vetoriais. No entanto, diversas aplicações científicas e de engenharia, como as da classe "*Grand Challenges*" [1], demandam uma capacidade de processamento hoje apenas vislumbrável com processamento paralelo de larga escala. Visando um melhor entendimento do comportamento destes programas de avaliação de desempenho em arquiteturas multiprocessadoras e como parte do desenvolvimento do multiprocessador **SPADE** [2,3], será feito um levantamento das características de localidade de referências e compartilhamento dos dados dos programas. Estes resultados serão importantes para a escolha de parâmetros arquiteturais como tamanho de bloco, tamanho de cache e protocolo de coerência de memória. Nas seções 2 e 3 faz-se uma apresentação dos conjunto **SPLASH** e do simulador **MINT**. Em seguida, na seção 4 são apresentados os resultados principais. Finalmente, na seção 5 tem-se as conclusões principais do trabalho.

2 - O CONJUNTO DE PROGRAMAS *SPLASH* DE STANFORD (EUA)

O **SPLASH** - "*Stanford Parallel Applications for Shared-Memory*" - [4] é um conjunto de programas aplicativos paralelos coletado pela Universidade de Stanford para uso em projeto e avaliação de sistemas multiprocessadores paralelos, distribuído livremente para fins acadêmicos. Os programas são paralelizados manualmente para o modelo de memória compartilhada utilizando os macros *parmacs* do Argonne National Laboratory.

3 - SIMULADOR

O simulador **MINT**[1] [5] é um simulador **SDP** (Simuladores Dirigidos à Programa) de domínio público desenvolvido na Universidade de Rochester, que permite a simulação de arquiteturas baseadas no processador R3000. Algumas das características do **MINT** são: execução eficiente através de processos leves[2] de **UNIX**, interpretação direta de programas objetos, sem a sobrecarga típica dos simuladores **SDE** (Simuladores Dirigidos à Execução) tradicionais, devido ao engordamento do programa com código adicional de simulação. Na seção 4 descreve-se a metodologia empregada nas simulações.

---

[1] MIPS Interpreter

[2] "threads"

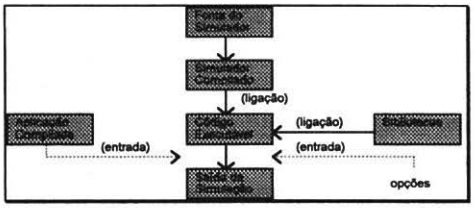

Fig. 1 - Esquema de Simulação com o MINT

4 - PROCEDIMENTO

Utilizando simuladores baseados no **MINT**, será feita uma avaliação do comportamento de algumas aplicações do **SPLASH**. Três características serão estudadas: ***i***) o custo da sincronização; ***ii***) a hierarquia de conjuntos de trabalho; e ***iii***) o número de cópias invalidadas em cada operação de escrita em caches que seguem a política de invalidação em escrita.

4.1 - HIERARQUIAS DE CONJUNTO DE TRABALHO

O conjunto de trabalho ("*Working Set*") é um conceito introduzido por Denning (1968), e representa o conjunto de endereços (ou páginas) referenciados por um programa durante uma dada janela de tempo [6]. No presente trabalho, têm-se interesse no conjunto de blocos de cache que um programa faz referência ao longo do tempo, e que portanto determina o tamanho que o cache deve ter para que haja um baixo índice de faltas em acesso ao cache. Cabe lembrar que o índice de faltas em acessos a cache em multiprocessadores depende, não apenas do tamanho em número de blocos, como também do nível de associatividade do cache (faltas relativas à conflito), do tamanho do bloco (se muito pequena, pode provocar faltas compulsórias, e se muito grande, provocar faltas devido a invalidações decorrentes de compartilhamento falso), e, naturalmente, das invalidações devido a compartilhamento real.

Estudos anteriores têm mostrado a existência de diversos pequenos conjuntos de trabalho e um grande que engloba toda a partição de dados do processador [7]. Em geral os conjuntos de trabalho menores apresentam maior impacto no desempenho do cache. A hierarquia de conjuntos de trabalho será determinada executando-se programas paralelos em uma arquitetura com barramento ideal, com caches totalmente associativos com diversos tamanhos de bloco. O barramento ideal permite serializar os acessos à memória e implementar um protocolo simples de coerência baseado em monitoração. O cache totalmente associativo permite eliminar as faltas devido à conflito.

A idéia consiste em se traçar o gráfico da taxa de faltas totais de cache em função do tamanho do cache para determinar os pontos de quebra da curva. Variando-se o

número de processadores e mantendo-se o tamanho do problema constante, pode-se ver como estes conjuntos de trabalho variam com o número de processadores.

O simulador, denominado "fullcache.c", é uma versão modificada do programa exemplo "cache.c" que acompanha o **MINT**. O programa "cache.c" é um simulador de arquiteturas multiprocessadoras com barramento, implementando caches infinitos de tamanho de bloco variável. Assumindo um cache separado para instruções e dados privados com tamanho suficientemente grande para prover um índice de acertos de praticamente 100%, as simulações avaliam apenas o desempenho do cache para acesso a dados (declarados como) compartilhados.

Os programas apresentam em geral uma hierarquia de conjuntos de trabalho composta de dois conjuntos de trabalho bem definidos (fig.2): ***i*) CT0** (Conjunto de Trabalho "0") - é o tamanho de cache suficiente para reduzir a taxa de faltas de 100% a patamar intermediário; ***ii*) CT1** (Conjunto de Trabalho "1") - é o tamanho de cache suficiente para reduzir a taxa de faltas a um nível "residual", e posteriores aumentos do tamanho do cache não alteram significativamemte este nível. Cabe destacar, todavia, que o comportamento descrito não é geral, e podem existir programas onde o patamar intermediário não é bem definido ou existir outras hieraquias.

Conhecido **CT1**, pode-se dividir a curva de taxa de faltas em função do tamanho do cache em duas regiões: i) Região faltas-substituição (tamanho de caches menores que **CT1)** - as faltas de substituição de bloco constituem a parte predominante da taxa de faltas; ii) Região faltas-invalidação (tamanho de cache maior ou igual a **CT1)** - as faltas devido a invalidação constituem a parte predominante da taxa de faltas. Como o cache a ser simulado é totalmente associativo, as faltas devido a conflito não ocorrem.

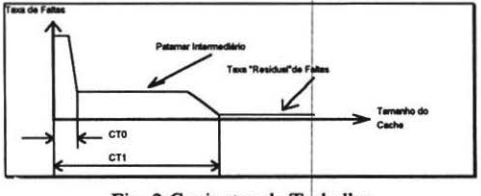

Fig. 2 Conjuntos de Trabalho

4.2 - NÍVEL DE COMPARTILHAMENTO

Ao longo da execução de um programa com variável compartilhada em um multiprocessador com caches coerentes, verifica-se que o número de compartilhadores (nível de compartilhamento) de um determinado bloco do memória compartilhada varia com o tempo. A cada novo leitor, a lista de compartilhadores é aumentada, até eventualmente atingir o número total de processadores no sistema. Quando ocorrer a

primeira operação de escrita ao bloco, a lista é invalidada, no caso do protocolo ser do tipo invalidação em escrita. Já se o protocolo for do tipo atualização em escrita, cada um dos caches da lista é atualizado, com o inconveniente de atualizar o cache de algum processador que já não mais precisa do dado.

Em protocolos de coerência de cache com lista ligada, como a empregada pelo padrão **IEEE/ANSI-SCI**, a sobrecarga de invalidação da lista pode comprometer a escalabilidade do sistema. Felizmente, estudos têm mostrado que o nível de compartilhamento típico de programas paralelos é baixo [8, 9], motivando o desenvolvimento de novos esquemas de diretórios baseados em diretório limitado [10,11,12].

Far-se-á aqui uma avaliação do número de compartilhadores utilizando uma versão instrumentada do programa simulador de caches infinitos "cache.c". Em cada operação de invalidação, o número de caches invalidados é avaliado para atualizar um histograma de compartilhamento. Como este número pode variar em função do tamanho do bloco, são feitas simulações para diversos tamanhos de bloco, em especial o tamanho de 64 bytes, que corresponde ao tamanho adotado pelo padrão **SCI**, e 4096, que corresponde ao tamanho típico de página.

A simulação com caches infinitos tem como objetivo evitar que o padrão de comportamento seja modificado em virtude de substituições de bloco ocasionados por faltas associadas à capacidade de caches finitos.

Diferente de outros estudos [13,14], onde o histograma de compartilhamento inclui invalidação de "0" cópias, no presente trabalho são consideradas apenas invalidações de 1 ou mais cópias, não incluindo o cache do próprio processador responsável pela invalidação pois este não provoca tráfego no sistema de interconexão.

5 - RESULTADOS

5.1 - PROGRAMA CHOLESKY

5.1.1 SINCRONIZAÇÃO

A figura (3.a) mostra o tempo médio que os processadores permaneceram bloqueados[3]. Para 128 processadores, em média cerca de 20 % do tempo total de execução é dispendido em estado de bloqueio em trava, o que explica em grande parte a baixa escalabilidade do programa para o tamanho de problema dado. Observa-se também um

[3] É o tempo médio de bloqueio em sincronização dividido pelo tempo total de execução substraído da tempo de inicialização sequencial. Na verdade este é um valor aproximado pois há um trecho sequencial após a fase paralela em que somente o processador "0" permaneceu trabalhando. Em especial no Cholesky, para problemas muito pequenos, este trecho pode ser significativo.

domínio do tempo de sincronização em semáforos em relação ao tempo de sincronização em travas. Os semáforos são utilizados no programa na implementação de barreiras, enquanto as travas são utilizadas apenas para proteger pequenas seções críticas. Calculando-se o tempo médio dispendido em semáforos, verifica-se que os processadores ficaram em média 77.256 ciclos (77 mil) esperando que todos os processadores que participam da barreira chegassem a ela, contra apenas 129 ciclos esperando que a trava fosse liberada (fig. 3.b). Este longo tempo dispendido em barreiras demonstra um pobre balanceamento de carga de trabalho.

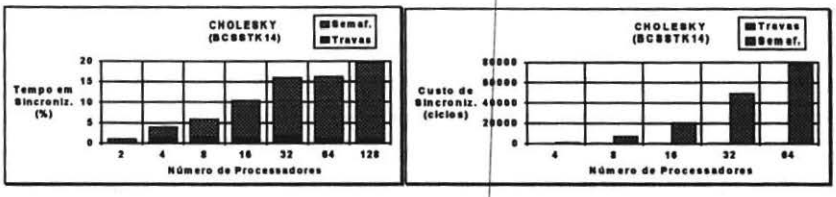

Fig. 3 CHOLESKY - a) Tempo Tot. de Sincroniz.; b) Custo Médio de Sincroniz.

5.1.2 HIERARQUIA DE CONJUNTOS DE TRABALHO

Nas figuras (4.a) e (4.b) tem-se gráficos da taxa de faltas em acessos ao cache para diversos tamanhos de cache (em número de blocos). As figuras mostram como a taxa de faltas em acessos de leitura e escrita, respectivamente, variam em função do número de blocos no cache e processadores no sistema, para tamanhos de blocos de 64 bytes.

Estes gráficos mostram que o tamanho ótimo de cache para o Cholesky não varia com o aumento do número de processadores. Um cache primário pequeno (menos de 1Kbytes) é suficiente para absorver a maioria dos acessos (taxa de faltas inferior a 20%), e um cache secundário não superior a 8Kbytes é suficiente como cache secundário.

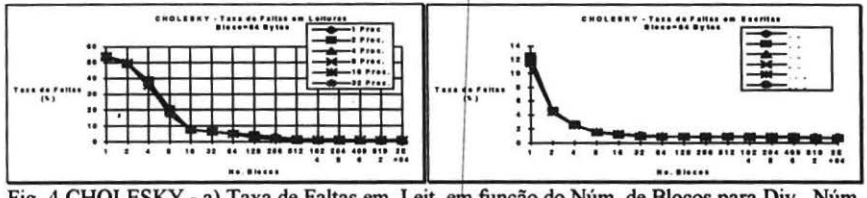

Fig. 4 CHOLESKY - a) Taxa de Faltas em Leit. em função do Núm. de Blocos para Div. Núm. de Proc.; b)Taxa de Faltas em Escr. em função do Núm. de Blocos para Div. Núm. de Proc.

Os gráficos das figuras 5 e 6 comparam caches com o mesmo número de blocos e com o mesmo tamanho em bytes para vários tamanhos de bloco. O intuito na comparação de caches de mesmo número de blocos é comparar caches implementados com tipos de memórias diferentes, como memórias estáticas rápidas e memórias dinâmicas lentas, que apresentam velocidade, custo por bit e capacidades diferentes. Como a parte de controle e memória de etiquetas dos caches necessita ser em geral implementada com circuitos de alta velocidade, é desejável que o número de blocos no cache não seja muito elevado. Para aumentar o tamanho do cache, limitando-se o número de blocos, uma forma é aumentar o tamanho de bloco: com o mesmo número de blocos, pode-se implementar caches pequenos com tamanho de bloco pequeno usando-se memórias estáticas, ou caches grandes com tamanho de bloco grande usando-se memórias dinâmicas. Já a comparação com tamanhos iguais em bytes é uma forma mais apropriada à comparação de caches com tecnologia de memória semelhantes.

A diminuição da taxa de faltas nas figuras (5.a) e (6.a) com o aumento do tamanho de bloco deve-se ao aumento do tamanho do cache e consequente redução das faltas associadas à substituição de blocos. De fato, nas figuras (5.b) e (6.b), pode-se ver que comparando caches de tamanhos em bytes iguais, os de tamanhos de bloco menores apresentam menor taxa de faltas.

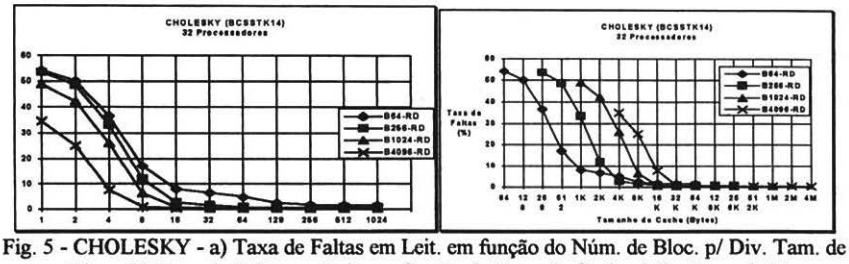

Fig. 5 - CHOLESKY - a) Taxa de Faltas em Leit. em função do Núm. de Bloc. p/ Div. Tam. de Bloco; b) Taxa de Faltas em Leit. em função do Tam. do Cachep/ div. Tam. de bloco

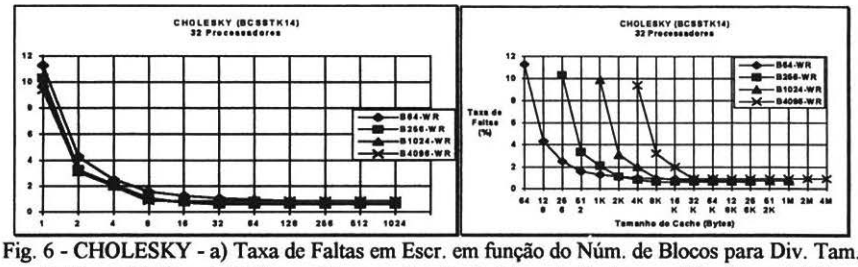

Fig. 6 - CHOLESKY - a) Taxa de Faltas em Escr. em função do Núm. de Blocos para Div. Tam. de Bloco; b) Taxa de Faltas em Escr. em função do Tam. do Cache para Div. Tam. de Bloco

5.1.3 - PADRÃO DE INVALIDAÇÃO

A figura (7.a) mostra o número de cópias invalidadas nas operações de escrita para diversos tamanhos de sistema, a saber, 4, 8, 16 e 32 processadores. Observa-se que o maior número de invalidações ocorre com 1 cópia, independentemente do número de processadores no sistema, mas o aumento do número de processadores aumenta o número de invalidações com mais cópias.

Algumas invalidações ocorrem com o número de cópias igual ao número de processadores do sistema (note os pontos em 7, 15 e 31), e se devem possivelmente a sincronizações globais, como barreira.

A figura (7.b) mostra que o número total de invalidações cresce com o aumento do número de processadores em decorrência do aumento do nível de compartilhamento (falso e verdadeiro). Por outro lado, vê-se na figura (7.c) que a distribuição do número de cópias invalidadas não se altera significativamente com o aumento de processadores. Nas figuras (7.b) e (7.c) pode-se ver que a maioria das invalidações de 1 cópia correspondem à invalidação de blocos já armazenados no cache do processador (elemento 1+ da legenda), e portanto, na verdade, blocos com 2 cópias: o próprio cache e o outro cache que será invalidado.

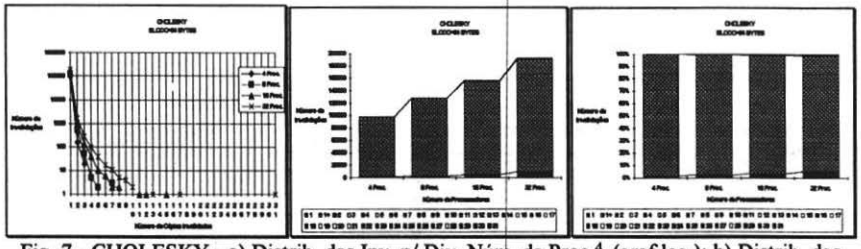

Fig. 7 - CHOLESKY - a) Distrib. das Inv. p/ Div. Núm. de Proc.[4] (graf.log.); b) Distrib. das Inv. p/ Div. Núm. de Proc.; c) Distrib. % das Inv. p/ Div. Núm. de Proc.

A figura (8.a) mostra como a distribuição das invalidações varia com o tamanho do bloco, mantido o número de processadores constante. No gráfico, variou-se o tamanho dos blocos de 4 bytes, que corresponde a uma palavra do processador (e possível tamanho de bloco da memória cache primária interna à pastilha do processador), à 4096, que corresponde ao tamanho típico de página dos processadores modernos. O tamanho de bloco de 4 bytes permite obter um gráfico de invalidações isento de compartilhamento falso, na medida em que todas as variáveis compartilhadas possuem tamanho iguais ou superiores a 32 bits.

---

[4] os valores não assinalados correspondem a 0 (zero) invalidações

A figura (8.b) mostra que a diminuição das invalidações com 1 cópia tem como efeito a redução do número total de invalidações. Na figura (8.c), vê-se que as invalidacões de 1 cópia tendem a cair em relação às demais.

As figuras (8.b) e (8.c) mostram também que o motivo da queda das invalidações de 1 cópia é a diminuição das invalidações de 1 cópia de blocos já armazenados no próprio cache (elemento 1+ das legendas). Para blocos de 4096 bytes, as invalidações de 1 cópia de blocos não armazenados no cache são majoritárias, revelando um comportamento tipo migratório: um bloco presente apenas em um outro cache está sendo invalidado para ser trazido a este. Este comportamento pode ser deletério para o desempenho se o número de reutilizações da cópia antes da migração for baixa. Como o bloco é grande, o custo associado à sua migração pode ser elevado fazendo com o que os processadores permaneçam muito tempo ociosos aguardando o término da migração. Além disso, blocos grandes podem comprometer o desempenho do sistema de interconexão.

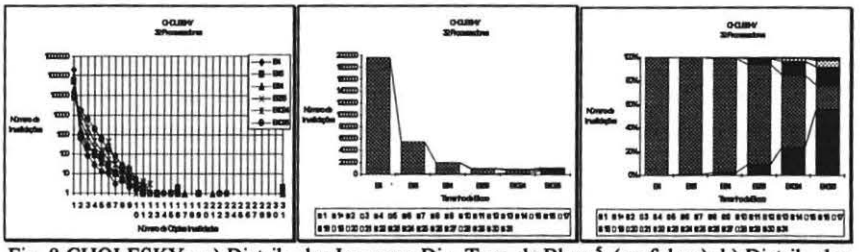

Fig. 8 CHOLESKY - a) Distrib. das Inv. para Div. Tam. de Bloco[5] (graf. log.); b) Distrib. das Inv. para Div. Tam. de Bloco; c) Distr. Percentual das Inv. para Div. Tam. de Bloco

5.2 PROGRAMA MP3D

5.2.1 SINCRONIZAÇÃO

A figura (9.a) mostra a percentagem média do tempo total dispendida em bloqueio pelos processadores para diversos tamanhos de sistema. Diferente do Cholesky, o tempo consumido em travas é significativo em relação ao tempo consumido em semáforos (barreiras).

A figura (9.b) mostra que apesar do tempo dispendido em travas e semáforos (barreiras) ser considerável, o tempo médio dispendido pelos processadores não é muito excessivo. No caso de travas, o tempo médio é 639 ciclos, e no caso de barreiras, o tempo médio é 1318 ciclos. Comparando com o tempo médio dispendido em barreiras

[5] Os valores não assinalados correspodem a 0 (zero)

no programa Cholesky, concluimos que o balanceamento de carga no MP3D é melhor do que no Cholesky.

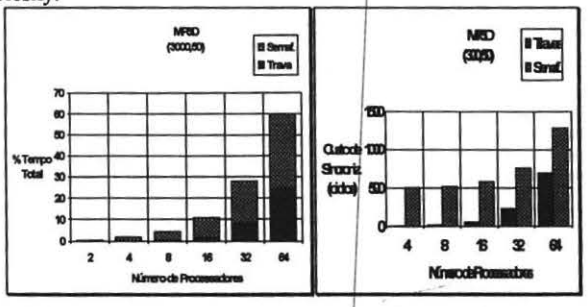

Fig. 9 - MP3D (3000,50) - a) Tempo Dispendido em Bloqueio (%); b) Custo Médio de Sincronização

5.2.2 HIERARQUIA DE CONJUNTOS DE TRABALHO

As figuras 10 e 11 mostram que o MP3D possui um grande Conjunto de Trabalho CT1. No entanto, assim como o CHOLESKY, para acessos de leitura, o CT1, mantido o tamanho do problema constante, decresce com o aumento do número de processadores. A taxa "residual" de faltas em leituras aumenta sensivelmente com o número de processadores, atingindo mais de 10% para 32 processadores, com cache de bloco de 32 bytes.

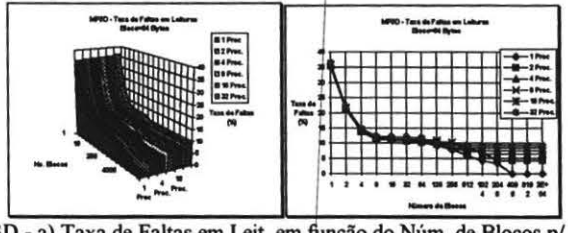

Fig. 10 - MP3D - a) Taxa de Faltas em Leit. em função do Núm. de Blocos p/ Div. Núm. de Proc.; b) Taxa de Faltas em Leit. em função do Núm. de Blocos p/ Div. Núm. de Proc.

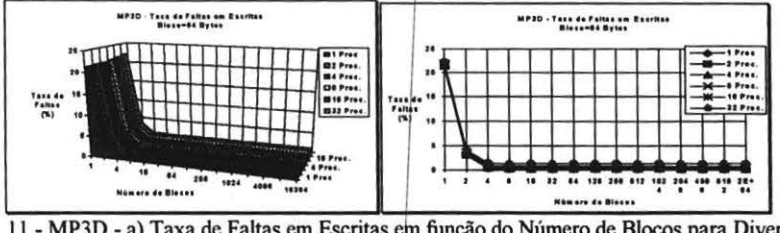

Fig. 11 - MP3D - a) Taxa de Faltas em Escritas em função do Número de Blocos para Diversos Números de Processadores; b) Taxa de Faltas em Escritas em função do Número de Blocos para Diversos Números de Processadores

Comparando-se caches com mesmo número de blocos (fig. 12 e 13) verifica-se que o aumento do tamanho de bloco não resulta em melhora apreciável da taxa de falta na região faltas-substituição, enquanto na região faltas-invalidação, para acessos de leitura, a taxa residual inicialmente cai com o aumento do tamanho de bloco mas depois passa a aumentar. Para caches com mesmo tamanho em bytes, na região de faltas-substituição, caches com tamanho de bloco maiores apresentam em geral taxa de faltas maiores.

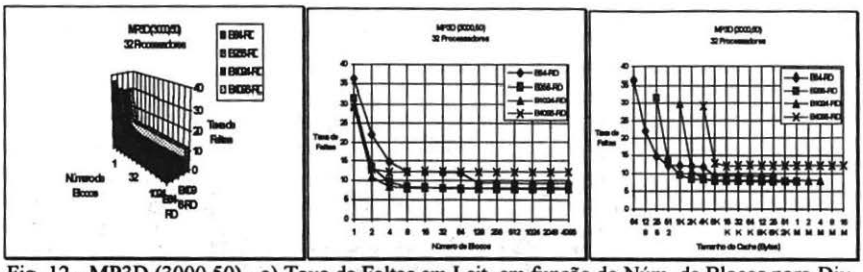

Fig. 12 - MP3D (3000,50) - a) Taxa de Faltas em Leit. em função do Núm. de Blocos para Div. Tam. de Bloco (graf. 3D); b) Taxa de Faltas em Leit. em função do Núm. de Blocos para Div. Tam. de Bloco; c) Taxa de Faltas em Leit. em função do Tam. do Cache para Div. Tam. de Bloco

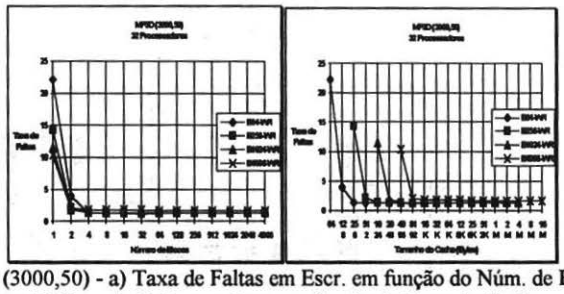

Fig. 13 - MP3D (3000,50) - a) Taxa de Faltas em Escr. em função do Núm. de Blocos para Div. Tam. de Bloco; b) Taxa de Faltas em Escr. em função do Tam. do Cache para Div. Tam. de Bloco

5.2.3 PADRÃO DE INVALIDAÇÃO

Diferente do Cholesky, o MP3D apresenta invalidações com o número de cópias invalidadas variando praticamente em toda a faixa (1 até o número de processadores), com o predomínio das invalidações de 1 cópia, como mostram as figuras 4.50, para diversos números de processadores, e 4.53, para diversos tamanhos de bloco.

Com o aumento do número de processadores (figura 14), mantido o tamanho de bloco constante, observa-se um aumento do número total de invalidações, mas a distribuição das invalidações não varia de forma significativa. As invalidações de 1 cópia

com acerto no cache permanecem como a parte predominante em relação à outras invalidações independente do número de processadores.

Aumentando-se o tamanho de bloco (figura 15), verifica-se uma redução do número total de invalidações até o tamanho de 256 bytes, com um ligeiro aumento a partir deste ponto. Aumentando-se o tamanho de bloco de 16 a 4096 byte observa-se um aumento relativo das invalidações de mais de 1 cópia e uma diminuição (relativa e absoluta) das invalidações de 1 cópia com falta em cache.

Weber et al. [14] mostram que a maioria das invalidações de duas ou mais cópias são devidas a acessos relacionados com sincronização e que dados compartilhados contribuem muito pouco para estas invalidações.

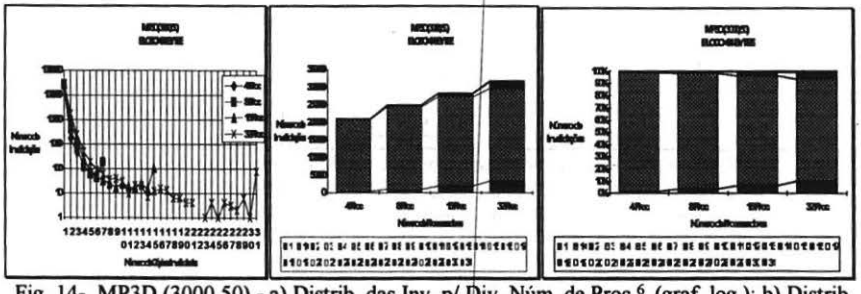

Fig. 14- MP3D (3000,50) - a) Distrib. das Inv. p/ Div. Núm. de Proc.[6] (graf. log.); b) Distrib. das Inv. p/ Div. Núm. de Proc.; c) Distrib. das Inv. p/ Div. Núm. de Proc.[7]

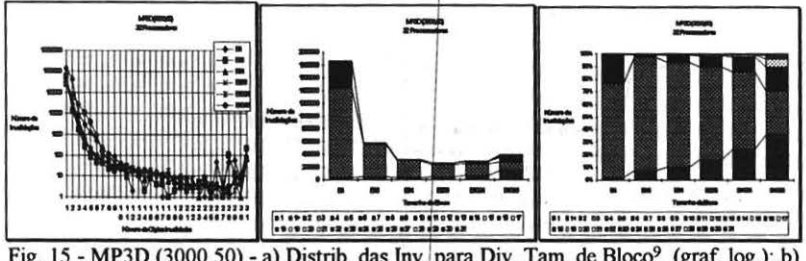

Fig. 15 - MP3D (3000,50) - a) Distrib. das Inv. para Div. Tam. de Bloco[9] (graf. log.); b) Distrib. das Inv. p/ Div. Tam. de Bloco; c) Distrib. % das Inv. p/ Div. Tam. de Bloco

6 - TRABALHOS RELACIONADOS

A caracterização de programas paralelos tem se constituido em tópico ativo de pesquisa visando não apenas uma melhor compreensão do comportamento e requisitos de programas paralelos reais, como também buscar novas soluções arquiteturais, melhor

[6] Os valores não assinalados correspondem a 0 (zero)
[7] Os valores não assinalados correspondem a 0 (zero)
[9]

moldados às características destes programas. Em especial, a característica de referência à memória tem ocupado uma posição de destaque em função do seu impacto em sistemas com memória compartilhada.

Um dos trabalhos mais importantes é devido a Egers & Katz [15], que propuseram a noção de "corrida de escrita" (em inglês "write-run") como forma de caracterizar o comportamento de compartilhamento.

Em outro trabalho importante, Agarwal & Gupta [16] analisaram o comportamento de compartilhamento de diversos programas executados sobre o sistema operacional **MACH**. Neste trabalho, analisam o comportamento dos programas segundo os conceitos de localidade temporal, espacial, e de processador. Weber & Gupta [14] analisam o padrão de invalidações em multiprocessadores, propondo uma classificação de objetos de dados segundo 5 categorias de compartilhamento: ***i***) código e dados apenas de leitura; ***ii***) objetos migratórios, ***iii***) objetos de sincronização, ***iv***) objetos predominantemente de leitura; e ***v***) objetos frequentemente de escrita/leitura. O trabalho mostra que o número de cópias invalidadas é frequentemente pequeno. Em particular, o programa **MP3D** é analisado visando o estudo de sua característica de invalidação, diferindo do presente pelo fato de se basear em análise de rastros de referência à memória obtidos através de método combinado de software/hardware em uma arquitetura específica, no caso o VAX-8350 e o VAX-3200. Ao contrário, o presente trabalho estuda o efeito do tamanho de bloco e número de processadores no padrão de invalidação.

Singh et al [7] fazem um minucioso estudo de hierarquia de conjuntos de trabalho, granularidade, e escalabilidade de diversos programas paralelos. Em particular, o programa Barnes-Hut é analisado visando determinar as características mencionadas. O artigo difere do presente trabalho nos seguintes pontos: ***i***) para determinar a hierarquia de conjuntos de trabalho, Singh et al. utilizam um simulador de caches com política de substituição **LRU**, ao invés da política **FIFO** adotada no presente trabalho em virtude de sua maior simplicidade de implementação; ***ii***) as simulações de Gupta et al. não variam o número de processadores ou o tamanho de bloco - como vimos, estes fatores podem afetar significativamente os resultados; ***iii***) as simulações consideram apenas os acessos de leitura - como vimos, o conjunto de trabalho **CT1** de acessos de escrita do Barnes-Hut é muito maior do que o conjunto de trabalho **CT1** de acessos de leitura; e ***iv***) não é claro no artigo se as simulações incluem somente os acessos compartilhados ou se eventualmente incluem os acessos não compartilhados.

7 - CONCLUSÕES

O presente trabalho estendeu os estudos anteriormente publicados sobre a característica de referência à memória dos programas Cholesky e MP3D, pertencentes ao conjunto SPLASH de Stanford.

Inicialmente, utilizando simulação dirigida à programa avaliou-se o tempo dispendido nas sincronizações. O estudo verificou que não somente o tempo total dispendido nas barreiras como também o tempo médio dispendido por processador em cada barreira, em especial no Cholesky, é elevado, o que pode ser sinal de fraco balanceamento de carga.

Em seguida, utilizando simulação dirigida à programa e mantendo-se o tamanho do problema constante, determinou-se a hierarquia de conjuntos de trabalho e sua dependência com o número de processadores no sistema e com o tamanho de bloco do cache. O aumento do número de processadores, uma vez que o tamanho do problema é mantido constante, resulta em uma diminuição do conjunto de trabalho CT1, com exceção do Cholesky.

Os resultados obtidos indicam que o tamanho de blocos/páginas em sistemas multiprocessadores com memória compartilhada é um importante parâmetro arquitetural, afetando a taxa de acertos e o número de invalidações. Uma arquitetura com tamanho ajustável de bloco/página selecionável a "run-time" pode ser uma solução a ser investigada no futuro.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] HWANG, Kai. "Parallel Processing Applications". In "*ADVANCED COMPUTER ARCHITECTURE - PARALLELISM, SCALABILITY, PROGRAMMABILITY*". McGraw-Hill, 1993, p. 118-129.

[2] Kofuji, S.T. et al. "*Projeto Hipersistemas: Uma Visão Geral*". Anais do Seminário de Supercomputação - Supercomp94. 1994.

[3] Kofuji, S.T. et al. "*O Multiprocessador SPADE*". Anais da III Jornada EPUSP-IEEE em Computação de Alto Desempenho. 1994.

[4] SINGH, J.P. et al.: "*SPLASH: Stanford Parallel Applications for Shared-Memory*". Computer Architecture News, v.20, n.1, p.5-44, March 1992.

[5] VEENSTRA, Jack. E. & FOWLER, Robert J."*MINT: A Front End for Efficient Simulation of Shared-Memory Multiprocessors*". Proceedings of the 2nd. International Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, p. 201-207, 1994.

[6] DENNING, P.J. "*Working Set Model for Program Behavior*". Communications of ACM, v. 11, n. 6, p. 323-333, 1968.

[7] SINGH, J.P. et al. "*Working Sets, Cache Sizes, and Node Granularity Issues for Large-Scale Multiprocessors*". Proceedings of the 20th Annual International Symposium on Computer Architecture, p.14-25, May 1993.

[8] AGARWAL, A. & GUPTA, A. "*Memory-reference characteristics of multiprocessor applications under MACH*". Proceedings of the ACM SIGMETRICS Conference on Measurements and Modeling of Computer Systems, p.215:225, 1988.

[9] CHAIKEN, David L. "*Cache Coherence Protocols for Large-Scale Multiprocessors*". Department of Electrical Engineering and Computer Science, MIT, 1990. MsC Thesis

[10] AGARWAL, A. et al. "*An evaluation of directory schemes for cache coherence*". Proceedings of the 15th International Symposium on Computer Architecture, p. 280-289, 1988.

[11] CHAIKEN, David et al. "*LimitLESS Directories: A Scalable Cache Coherence Scheme*". Proceedings of the 4th International Conference on Architectural Support for Programming Languages and Operating Systems, p. 224-234, April 1991.

[12] WOOD, David A. et al. "*Mechanisms for Cooperative Shared Memory*". Proceedings of the 20th Annual International Symposium on Computer Architecture, p. 156-167, 1993.

[13] THAPAR, Manu. "*CACHE COHERENCE FOR SCALABLE SHARED MEMORY MULTIPROCESSORS*". Computer Systems Laboratory, Stanford University, Tech. rep. CSL-TR-92-522, May 1992. PhD Thesis.

[14] WEBER, Wolf-Dietrich & GUPTA, Anoop. "*Analysis of Cache Invalidation Patterns in Multiprocessors*". Proceedings of the 3rd. International Conference on Architectural Support for Programming Languages and Operating Systems, p. 243-256, April 1989.

[15] EGGERS, Susan & KATZ, Randy H. "*A Characterization of Sharing in Parallel Programs and its Application to Coherence Protocol Evaluation*". Proceedings of the 15th Annual International Symposium on Computer Architecture, p. 373-382, 1988.

[16] AGARWAL, A. & GUPTA, A. "*Memory-reference characteristics of multiprocessor applications under MACH*". Proceedings of the ACM SIGMETRICS Conference on Measurements and Modeling of Computer Systems, p.215:225, 1988.