ANÁLISE DE DESEMPENHO DE UM BARRAMENTO ESPECIAL PARA SINCRONIZAÇÃO DE BARREIRA EM MÁQUINAS PARALELAS DE MEMÓRIA COMPARTILHADA

Martha X.T.Delgado, Edward D.M.Ordoñez, Sergio T.Kofuji [1]

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

RESUMO

Neste artigo apresentamos uma análise quantitativa do comportamento de um barramento especial para sincronização de barreira em uma máquina paralela de memória compartilhada baseada em barramento. Para realizar este estudo desenvolvemos um modelo analítico e um modelo de simulação. Com estes modelos avaliamos a sobrecarga de tempo gerada pelo barramento em diferentes condições de carga da máquina paralela e comparamos seu desempenho com outra solução em software.

ABSTRACT

In this article we present a quantitative analysis of barrier synchronization bus behavior in a bus-based shared-memory multiprocessor. In order to carry out this study we develop analytical and simulation models. With these models we evaluate the overhead arising from barrier synchronization bus for a variety of workloads on a target multiprocessor system and we compare barrier synchronization bus performance with other software solution.

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

1. INTRODUÇÃO

Para realizar sincronização de barreira de maneira eficiente geralmente é necessário implementar a sincronização de barreira em *hardware*. Na literatura existem muitas soluções implementada em *hardware* [1], dentro destas soluções existem aquelas que são implementadas em um barramento especial. Algumas destas soluções usam o barramento para realizar outras operações de sincronização [2], ou operações de escalonamento [3], outras tentam fazer o barramento mais flexível para realizar diferentes classes de sincronização de barreira [4] [5] [6]. Em geral a característica comum das soluções implementadas em um barramento especial é a economia em custos.

Nosso barramento especial para sincronização de barreira proposto em [4] é uma solução econômica, eficiente e flexível, embora não apresente características de escalabilidade. Permite fazer a sincronização de qualquer número de processadores, permite sincronizar concorrentemente várias barreiras e permite a execução de barreiras diferentes consecutivas. Pode ser usada tanto para sincronizar comandos como para sincronizar laços.

Neste trabalho mostramos uma análise quantitativa do comportamento do barramento especial em uma máquina paralela de memória compartilhada baseada em barramento. A análise é feita mediante modelos analíticos e de simulação que nos permitem avaliar a sobrecarga de tempo gerada pelo barramento especial em diferentes condições de carga da máquina paralela.

No seguinte item vamos explicar de maneira sucinta em que consiste nosso barramento especial de sincronização de barreira e o modelo analítico desenvolvido para avaliar seu desempenho, no item 3 mostramos uma comparação analítica do desempenho entre nosso barramento especial e a melhor solução de *software* para sincronização de barreira em uma máquina de memória compartilhada baseada em barramento. No item 4 mostramos o comportamento do barramento especial em uma situação real mediante um modelo de simulação. No item 5 mostramos uma comparação entre o modelo analítico e o modelo de simulação. Finalmente no item 6 apresentamos algumas conclusões do trabalho.

2. FUNCIONAMENTO BÁSICO DO BARRAMENTO ESPECIAL

Na figura 1 mostramos nossa solução a qual consta de um barramento especial de sincronização e um bloco de circuitaria (bi) adicional para cada processador (Pi). No momento que um processador chega na barreira emite através do barramento o nome de sua barreira. Os demais processadores emitem uma resposta e o bloco de circuitaria realiza a sincronização de barreira.

O barramento especial é usado para realizar a comunicação entre os blocos de circuitaria, nele emite-se o nome da barreira do processador, e um sinal chamado de SB (sincronização de barreira) o qual é usado para emitir a resposta de sincronização de barreira, este é um fio *"wired-or"*. O bloco de circuitaria contem dois registradores de sincronização importantes os quais são: NB onde está o nome da barreira na qual o processador sincroniza-se, e CH1 que é um registrador de 1 bit no qual o processador avisa no momento que chega na barreira.

No momento que o processador (P_i) chega na barreira, o bloco de circuitaria (bi) acesa o barramento (existe um protocolo de arbitração para dar acesso ao barramento a um bloco de circuitaria por vez, neste caso usa-se um arbitrador central (AC na figura 1)), emite o nome de sua barreira nele, e põe um 0 na linha sincronização de barreira (SB). Ao mesmo tempo os blocos de circuitaria dos demais blocos de circuitaria comparam o nome da barreira no barramento com o valor que está em seu registrador NB. Se os dois valores são iguais e o processador chegou então o bloco de circuitaria é encarregado de por um 0 na linha SB. Se são iguais e o processador ainda não chegou então o bloco de circuitaria põe um 1 na linha SB. E Se são diferentes o bloco de circuitaria põe um 0 na linha SB. Depois de um tempo, cada bloco de circuitaria lê o estado da linha SB, se a linha está em 0 então todos os processadores que se estão sincronizando nessa barreira sabem que a sincronização foi feita e se está em 1 significa que ao menos um processador não chegou ainda na barreira então o bloco de circuitaria tenta de novo pegar o barramento.

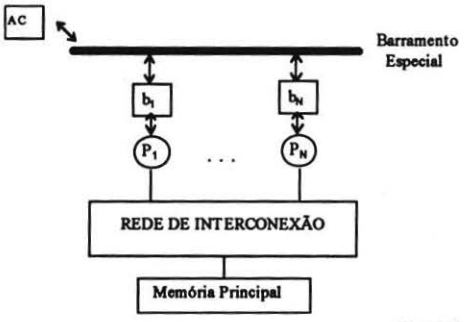

Figura 1. Diagrama do barramento especial para sincronização de barreira.

O barramento especial precisa de $\log_2(P-1)$ linhas para pôr o nome da barreira, com este número pode suportar P - 1 barreiras simultâneas (P é o número de processadores). Também precisa de 3 linhas de controle (para o protocolo de arbitramento) e a linha SB. Em total o número de linhas do barramento especial de sincronização de barreira é $\log_2(P-1) + 4$.

2.1. Descrição do Modelo Analítico

O modelo analítico foi desenvolvido com o objetivo de analisar o comportamento e o desempenho da solução em diferentes situações. Particularmente o modelo analisa o desempenho da solução quando há uma sincronização de barreira no tempo. Consideramos N processadores no sistema com P processadores participando na sincronização de barreira.

O modelo analítico calcula o tempo de sincronização de barreira que está definido como o tempo que começa a contar desde o momento que o último processador chega na barreira até que todos os processadores saibam que todos chegaram, este tempo mede o desempenho da solução. O modelo proporciona dois resultados o limite superior e o limite inferior. O limite superior está definido como o pior caso e o limite inferior como o melhor caso. Portanto o comportamento genérico da solução sempre deve estar dentro desses limites.

A expressão (1) mostra o tempo de sincronização de barreira no melhor caso:

$$T(ns) = 25 + 3x\,0.086\,(N-1)\sqrt{1+13.3\,[N/(N-1)]} \quad (1)$$

A expressão (2) mostra o tempo de sincronização de barreira no pior caso:

$$T\,(ns) = (N - P + 1)(13 + 2x\,0.086\,(N-1)\sqrt{1+6.65\,[N/(N-1)]}) + 3x\,0.086\,(N-1)\sqrt{1+13.3\,[N/(N-1)]} + 24.5 \quad (2)$$

Estas expressões foram calculadas para as seguintes propriedades do barramento: tecnologia ECL F100K de alta velocidade e alto desempenho, e linhas do barramento par trançado AWG-26. Consideramos também uma máquina paralela de memória compartilhada de N processadores e P processadores participando da sincronização.

3. COMPARAÇÃO ANALÍTICA DO DESEMPENHO DO BARRAMENTO ESPECIAL

Para estabelecer uma comparação do desempenho apresentado pelo barramento especial é necessário quantificar a sobrecarga de tempo gerada por outras soluções em *software*. Escolhemos uma máquina paralela de memória compartilhada baseada em barramento para ser o alvo das medidas principalmente porque o barramento especial é uma solução eficaz para máquinas de pequeno e mediano porte (como as máquinas em menção). A solução em *software* que melhor desempenho apresenta nestas máquinas é o contador central porque gera menos operações sobre o barramento de dados[7].

O contador central é a solução tradicional: consiste em usar um contador central compartilhado, onde cada processador que chega à barreira incrementa o contador na memória compartilhada (o contador inicialmente tem o valor zero), e fica esperando em um laço ("espera em ciclo" (*"spin waiting"*)) até que o contador central chegue ao valor de N (número total de processadores participando na sincronização de barreira). Após isso, todos os processadores são liberados.

Com o objetivo de que o contador central apresente um melhor desempenho, supomos que na máquina alvo, a primitiva de sincronização busca-e-soma (BeS) (*"F&A"*) é implementada em *hardware*, também que apresenta um protocolo de coerência cache *snoopy* usando atualização de escrita (*"write-update"*). Com estas características a "espera em ciclo" do algoritmo contador central é feita na cache de cada processador.

3.1. Modelo Analítico para o Contador Central

Primeiro consideramos que todos os processadores chegam ao mesmo tempo na barreira, isto é chamado de carga balanceada. O tempo de sincronização de barreira para o contador central é:

$$T = P x T_{BeS} + T_{\text{escrita do bloco}} \quad (3)$$

O primeiro termo corresponde à fase de entrada do algoritmo do contador central, onde cada processador que chega na barreira deve executar atomicamente uma instrução BeS para incrementar o valor do contador central. O último termo corresponde à fase de saída do algoritmo, onde o último processador que chega na barreira faz uma escrita numa variável

compartilhada que gera um acesso ao barramento para que o bloco seja atualizado em todos os processadores [7]. Consideramos que a operação BeS consome 4 ciclos do barramento de dados e o custo de uma escrita para uma variável compartilhada no protocolo atualização de escrita depende da largura da banda do barramento de dados e do tamanho do bloco do cache; com um tamanho do bloco de 16 bytes e uma largura de banda de 8 bytes tem-se que este custo é 4 ciclos do barramento de dados, onde os 2 ciclos iniciais corresponde ao endereço. Tudo isto baseado nos dados da máquina Alliant FX/8 [3]. Estes tempos somente estão considerando a latência do barramento e não os tempos de acesso às memórias. Também consideramos que durante a sincronização de barreira não existe outro tráfico na rede.

Quando os processadores não chegam ao mesmo tempo na barreira chama-se de carga desbalanceada. Neste caso consideramos que o último processador que chega na barreira encontra que todos os processadores culminaram a fase de entrada, então o tempo de sincronização de barreira para o contador central é:

$$T = 1xT_{BeS} + T_{escrita\ do\ bloco.} \qquad (4)$$

O primeiro termo corresponde ao último processador fazendo sua operação de BeS e o último termo é o mesmo da expressão (3).

Na figura 2 apresentamos a diferença entre o tempo de sincronização de barreira do contador central e do barramento especial para sincronização de barreira, no caso de carga balanceada. Neste caso o comportamento do contador central é linear e bastante maior do que aquele do barramento especial. Na máquina paralela com barramento especial não se tem contenção no barramento de dados quando todos os processadores chegam ao mesmo tempo na barreira.

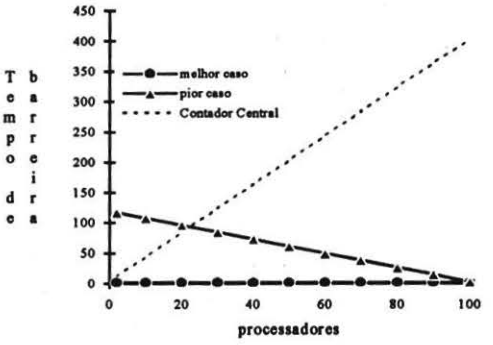

Figura 2. Tempo de sincronização de barreira para o contador central e o barramento especial, considerando carga balanceada.

Na figura 3 ilustramos a diferença das duas soluções quando temos carga desbalanceada. Observamos que o tempo no melhor caso de nossa solução é sempre menor do que o tempo da solução em *software*. Agora, no pior caso, a diferença vai diminuindo com o acréscimo do número de processadores participando na sincronização de barreira. É necessário salientar que o

tempo de sincronização de barreira, no pior caso, é relativamente maior devido principalmente ao tipo de arbitramento usado no barramento de sincronização. Se fosse usado um outro mecanismo de arbitração com certeza este tempo diminuiria significativamente. Em geral o barramento especial apresenta um bom comportamento tanto para cargas balanceadas quanto desbalanceadas.

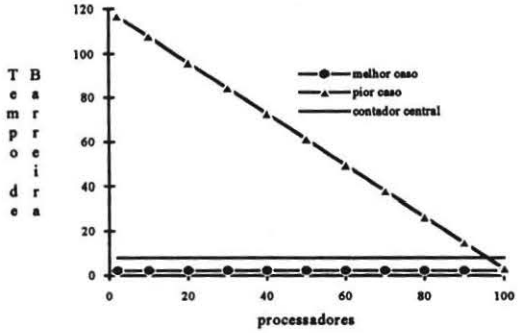

Figura 3. Tempo de sincronização de barreira para o contador central e o barramento especial, considerando carga desbalanceada.

4. RESULTADOS DE SIMULAÇÃO

Com o objetivo mostrar o comportamento de nossa solução em uma situação mais real desenvolvemos um modelo de simulação para avaliar o desempenho do barramento especial em uma máquina paralela específica que executa programas de aplicação. A simulação é feita usando o simulador MINT [8]. MINT é um programa de simulação acionado por programa (*"program driven"*) que permite construir simuladores de diferentes arquiteturas de multiprocessadores. Ele permite simular qualquer máquina avaliando seu comportamento quando está executando um determinado programa paralelo de aplicação.

4.1. Descrição Básica do MINT

O MINT consiste em duas partes principais: um **gerador de referência de memória** que modela a execução de um programa de aplicação e um **simulador do sistema alvo** que simula a arquitetura desejada.

O gerador de referência de memória modela a execução de um programa de aplicação. Para nosso caso usamos alguns dos programas de aplicação do SPLASH [9]. Quando o programa de aplicação produz uma operação que precise do uso da rede de interconexão ou da memória global este envia um evento ao simulador do sistema alvo. O simulador do sistema alvo simula a realização do evento baseando-se na arquitetura desejada e, quando as operações relativas ao evento acabam, é enviado um sinal para o gerador de referência de memória indicando que pode continuar.

Por exemplo, quando o programa de aplicação gera uma leitura, o MINT gera um evento de leitura e chama a função **sim_read()** (esta função deve estar definida no simulador do sistema

alvo para poder descrever o comportamento da arquitetura da máquina). Igualmente, quando ocorre um evento de escrita MINT chama a função **sim_write()**. No caso específico da sincronização de barreiras temos o seguinte: quando um processo chega na barreira MINT chama a função **sim_barrier_attempt()** e quando um processo passa a barreira MINT chama a função **sim_barrier_acquire()**. O valor de retorno destas funções controlam a execução do processo: se o valor que retorna é 0 permite que o processo que gerou o evento continue a execução e se é 1 então o processo no pode continuar executando instruções.

Para simular o barramento especial dentro do simulador MINT é necessário definir completamente o tipo de máquina que será usada. A máquina de memória compartilhada usada é uma máquina baseada em barramento. Cada processador tem associado uma *cache* infinita, a qual obedece um protocolo de coerência que consiste em invalidação de escrita (*"write invalidate"*), com um tamanho de bloco de 16 bytes. Uma falha de leitura (*"miss"*) produz um acesso ao barramento de dados. Uma falha de escrita produz um pedido de leitura exclusivo ao barramento de dados. Finalmente, um sucesso de escrita (*"hit"*) a um bloco compartilhado produz um pedido de invalidação ao barramento de dados.

Para implementar o barramento de sincronização de barreira usamos as funções **sim_barrier_attempt()** e **sim_barrier_acquire()**. Quando um processador chega na barreira, o MINT chama a função **sim_barrier_attempt()**, cada vez que esta função é chamada deve-se emitir um pedido de acesso ao barramento especial. Todas as operações de acesso ao barramento são feitas através da função barramento_mxtd() a qual simula o comportamento do mesmo. Na figura 4 mostramos o diagrama de fluxo da função barramento_mxtd()

Inicialmente o barramento avalia se está ocupado ou não. Se o barramento estiver desocupado então avalia-se se é a vez do processador assumir o barramento, isto é, simula-se o protocolo de arbitramento. Caso seja sua vez de assumir o barramento então a função incrementa o tempo de acesso a este e avalia se todos chegaram. Caso não chegaram então tenta de novo assumir o barramento. Senão for a vez do processador, este deve avaliar se todos chegaram com o objetivo de saber se é o último processador a chegar na barreira. Caso seja o último deve atualizar o tempo T pois a sincronização de barreira será feita logo após a vez de qualquer um dos processadores que participam da sincronização de barreira.

Senão for o último processador a chegar na barreira então tenta de novo ganhar o barramento. É necessário salientar que o trecho de código correspondente ao protocolo de arbitramento na função barramento_mxtd() sempre avalia a possibilidade que o barramento seja assumido por qualquer um dos processadores que chegou na barreira e está tentando pegar o barramento.

O MINT gera a função **sim_barrier_acquire()** quando todos os processadores chegam na barreira. O tempo de chegada deste evento e inicializado com o tempo que demorou em realizar a sincronização de barreira e que está armazenado na variável T

A estrutura de dados do MINT, na qual encontram-se armazenados os estados dos blocos da memória cache da máquina simulada, está limitada no uso por 32 processadores. Portanto as medidas feitas com o simulador usam até 32 processadores. O MINT roda em estações de

trabalho baseadas em MIPS incluindo as da Silicon Graphics e da Digital Equipment Corporation. O código fonte do MINT esta disponível via anonymous ftp desde cs.rochester.edu incluindo tutorial e manual do usuário. A seguir vamos explicar as características básicas dos programas de aplicação usados.

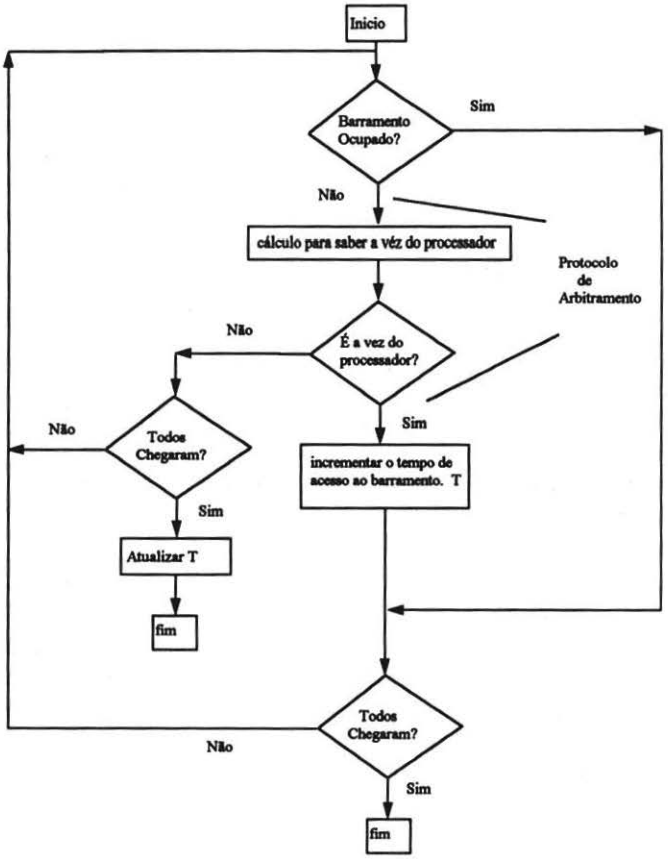

Figura 4. Diagrama da função barramento_mxtd ().

4.2. Descrição Básica dos Programas de Aplicação

O SPLASH é um conjunto de programas de aplicação usados no projeto e avaliação de sistemas multiprocessadores de memória compartilhada [9]. O SPLASH é feito para medir o desempenho de máquinas paralelas pequenas e de mediano porte. Cada programa apresenta um conjunto de dados de entrada que foram produzidos levando em conta esta consideração.

Estes programas de aplicação foram feitos sob o mesmo modelo de programação e independentes da arquitetura. A maioria dos programas são escritos em C. Tipicamente o processo pai divide-se em um determinado número de filhos que corresponde a um por

processador adicionado. O tipo de escalonamento usado geralmente é o escalonamento estático. As operações de sincronização usadas são *"locks"* e barreiras.

Para avaliar o desempenho do barramento especial utilizamos 3 programas do SPLASH : Water, Barnes-Hut e MP3D. Na tabela 1 mostramos as características gerais de cada uma destas aplicações. Na tabela 2 mostramos algumas características básicas de cada programa.

Tabela 1. Características gerais dos programas de aplicação.

Aplicação	Autor	Campo de Ação	Função
Water	J.P.Singh	dinâmica molecular	simula a evolução de um sistema de moléculas de água
Barnes-Hut	E. Bruni, D.Roger, J.P.Singh	astrofísica	simula a evolução das galáxias
MP3D	J.D.McDonald	aeronáutica	simula o fluxo de fluidos de baixa densidade

Tabela 2. Características básicas de cada programa de aplicação.

Aplicação	Linguagem	Linhas	Tipo de Escalonamento	Tipo de Sincronização	Granularidade
Water	C	1500	estático	barreiras	grossa
Barnes-Hut	C	2750	dinâmico : programador	barreira e bandeiras	grossa
MP3D	C	1500	estático	barreiras	grossa

4.3. Desempenho do Barramento Especial

Para cada programa de aplicação realizamos dois tipos de simulações, a primeira usando o barramento especial e a segunda usando o algoritmo de *software* contador central. As simulações foram feitas para P=2 até P = 32 processadores. O MINT permite saber em quanto tempo foi realizada a sincronização de barreira em cada uma das barreira dos programas. As unidades de tempo vêm dadas em ciclos de relógio do processador.

Na figura 5 mostramos o tempo de sincronização de barreira quando usamos o barramento especial para a primeira e a última barreira do programa Water. O Water é um programa de dinâmica molecular que avalia as forças e os potenciais em um sistema de moléculas de água no estado líquido. O programa usa em total 10 barreiras, realiza cálculos em N passos, onde N é fornecido pelo usuário. Os parâmetros de entrada usados nas simulações foram número de moléculas = 128 e número de passos = 2.. Cada rodada do MINT demorou em média 20 minutos.

As mudanças de tempo apresentadas na figura 5 obedecem ao protocolo de arbitramento usado no barramento especial. Por exemplo quando participaram 10 processadores na primeira barreira, o último processador chegou no momento que era sua vez para pegar o barramento e portanto não se gera sobrecarga para esperar a realização da sincronização de barreira. Já no caso da última barreira o último processador em chegar esperou para pegar o barramento e finalizar a sincronização.

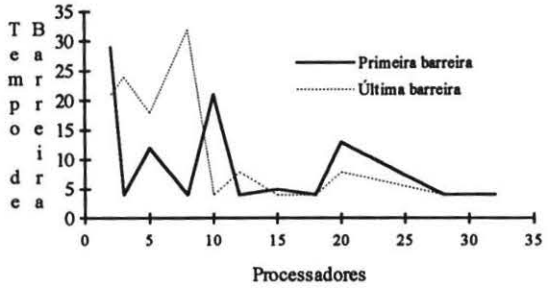

Figura 5. Tempo de sincronização de barreira para a primeira e a última barreiras do Water usando o barramento especial.

Na figura 6 mostramos o tempo de sincronização de barreira quando usamos o barramento especial e quando usamos o algoritmo contador central (é necessário salientar que este tempo somente leva em conta o tempo da fase de entrada do algoritmo e despreza o tempo de "espera em ciclo" e o tempo da fase de saída do algoritmo) para a primeira barreira do programa Water.

O tempo de sincronização de barreira no barramento é significativamente menor e não se afeta com o acréscimo de número de processadores participando da sincronização de barreira.

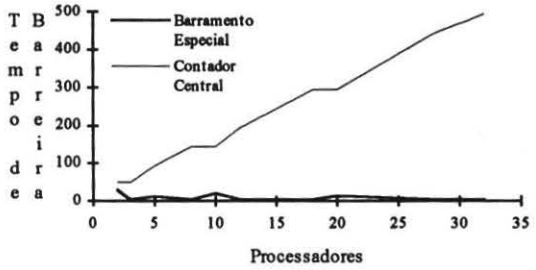

Figura 6. Comparação do tempo de sincronização de barreira na primeira barreira do Water, quando se usa o barramento especial e o algoritmo contador central.

Na figura 7 mostramos o tempo de sincronização de barreira para o barramento especial e para o algoritmo contador central, na primeira barreira do programa Barnes-Hut. Barnes-Hut simula a evolução de um sistema de corpos sob a influenza das forças gravitacionais. Cada corpo é modelado como uma massa pontual que exerce uma força nos demais corpos do sistema. A simulação é feita por passos, em cada passo calcula-se a força bruta em cada corpo e atualiza-se sua posição e outros atributos. Barnes-Hut apresenta um arquivo de entrada que contem os parâmetros de entrada. Nas simulações realizadas usaram-se 128 corpos. Foram usadas 102 barreiras. Cada rodada do MINT demorou em média 30 minutos.

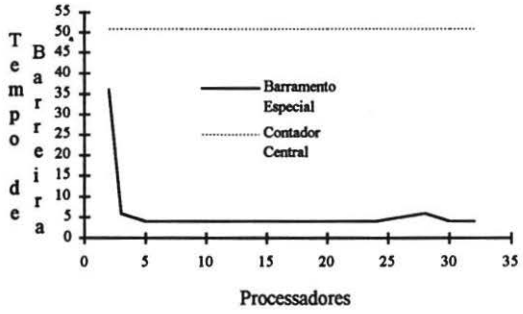

Figura 7. Comparação do tempo de sincronização de barreira na primeira barreira do Barnes-Hut quando se usa o barramento especial e o algoritmo contador central.

Neste caso observamos que o tempo de sincronização de barreira é sempre menor e que o comportamento do contador central já não é linear devido a que o balanceamento de carga entre o Barnes-Hut e o Water é diferente, como mostramos no item seguinte.

4.4. Análise do Efeito da Distribuição de Carga no Desempenho da Solução

Como mostramos na figura 6 o tempo de sincronização de barreira para o algoritmo contador central no programa Water apresenta um comportamento linear. Isto devido à distribuição da carga que apresentam as tarefas que estão sendo executadas pêlos processadores que vão a participar da sincronização de barreira. Na figura 8 mostramos os tempos de chegada de cada um dos processadores na primeira barreira do Water quando estão participando 28 processadores na sincronização de barreira e na figura 9 mostramos a mesma gráfica ampliada nos últimos 500 ciclos.

Neste caso muitos processadores chegam ao mesmo tempo na barreira gerando contenção no barramento de dados, que faz com que o algoritmo contador central apresente um comportamento linear. Observamos que a distribuição de carga não afeta o comportamento do barramento especial.

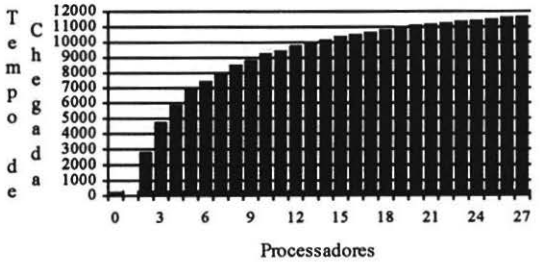

Figura 8. Distribuição da carga para a primeira barreira no programa Water quando participam 28 processadores.

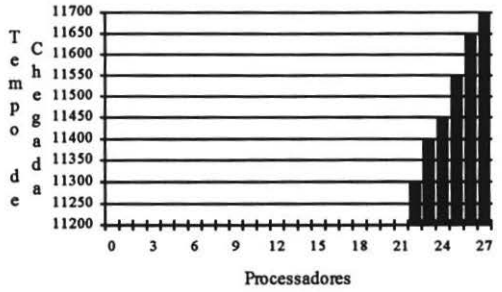

Figura 9. Distribuição de carga para a primeira barreira no programa Water quando participam 28 processadores (ampliando a escala de tempo).

No programa Barnes-Hut a distribuição de carga é menos densa como mostramos na figura 10, na qual encontramos os tempos de chegada de cada um dos processadores quando estão participando 28 processadores na sincronização de barreira.

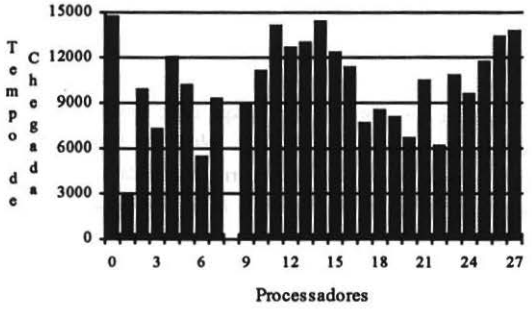

Figura 10. Distribuição da carga para a última barreira no programa Barnes-Hut quando participam 28 processadores.

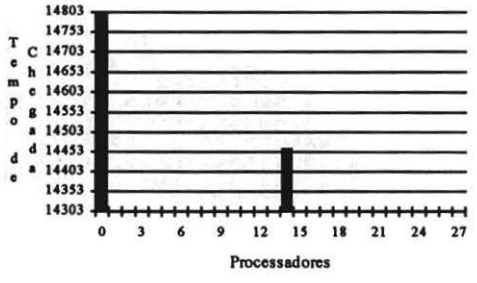

Figura 11. Distribuição de carga para a última barreira no programa Barnes-Hut quando participam 28 processadores (ampliando a escala de tempo).

Na figura 11 ampliamos a figura 10 para os últimos 500 ciclos, observamos então que o último processador que chegou na barreira está praticamente sozinho para assumir o barramento

de dados e executar a fase de entrada do algoritmo. Portanto o comportamento do contador central já não é linear devido a que não tem congestão para acessar o barramento no final da execução da barreira. Observamos também que o comportamento do barramento especial é independente da distribuição de carga usada. Na figura 7 mostramos que igualmente não afeta a distribuição de carga.

Observamos também que o algoritmo contador central apresenta melhor desempenho quando o desbalance de carga é grande. Isto corrobora a análise feita no item 3.

5. COMPARAÇÃO ENTRE O MODELO ANALÍTICO E DE SIMULAÇÃO

Para estabelecer uma comparação entre o modelo analítico e os resultados obtidos com o MINT, fazemos equivalente a expressão (1) (que permite determinar o tempo de sincronização de barreira do barramento especial no melhor caso) ao tempo de melhor caso considerado no MINT. Portanto avaliando a expressão (1) para N = 32 processadores temos que Tmelhor caso = 63.19 ns é equivalente a 4 ciclos do processador.

A expressão (2) determina o tempo de sincronização de barreira para o pior caso; avaliando-a para N = 32 processadores temos:

$$T_{\text{pior caso}} = 61.59 + 55.4 \times (33 - P) \qquad (6)$$

Na tabela 3 ilustramos a equivalência feita para ciclos de processador. Nas figuras 12, 13, e 14 mostramos o tempo de sincronização de barreira para o pior e o melhor caso correspondente ao modelo analítico quando comparado com cada uma das aplicações, para a primeira barreira.

Tabela 3. Tempo de sincronização de barreira para o pior caso.

Processadores	2	3	5	8	10	12	15	18	20	24	28	30	32
Tempo de Barreira Pior caso (ciclos)	113	110	103	92	85	78	68	57	50	36	22	15	8

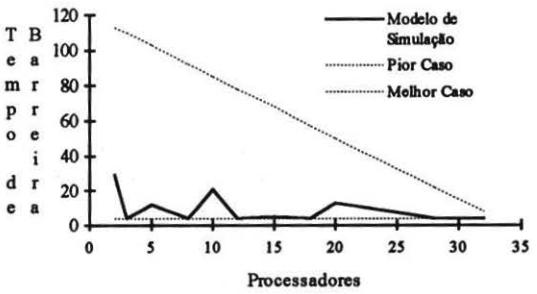

Figura 12. Comparação entre o modelo de simulação e o modelo analítico para a primeira barreira do programa Water.

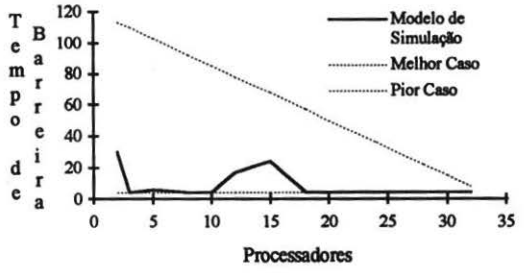

Figura 13. Comparação entre o modelo de simulação e o modelo analítico para a primeira barreira do programa Barnes-Hut.

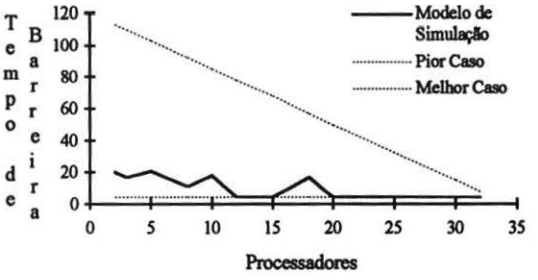

Figura 14. Comparação entre o modelo de simulação e o modelo analítico para a primeira barreira do programa MP3D.

Como dissemos no item 2, o modelo analítico proporciona os valores limites do comportamento da solução. O comportamento do modelo de simulação encontra-se sempre dentro dos limites dos cálculos feitos com o modelo analítico. O tempo de sincronização de barreira segundo o modelo de simulação sempre foi menor do que o pior caso no modelo analítico.

6. CONCLUSÕES E TRABALHOS FUTUROS

O barramento especial para sincronização de barreira é uma solução alternativa para máquinas de pequeno e mediano porte que oferece flexibilidade e eficiência. Seu desempenho é praticamente independente da distribuição de carga usada na máquina entanto que depende principalmente do tipo de mecanismo de arbitramento usado. Incrementando o número de linhas no barramento poderia ser projetado outro tipo de mecanismo de arbitramento que permitiria obter um melhor desempenho mas que afetaria a economia do barramento.

As simulações realizadas permitem concluir que o barramento especial gera pouca sobrecarga de tempo e seu desempenho é independente da distribuição de carga que apresente a aplicação.

Infelizmente nenhuma aplicação do SPLASH tem barreiras simultâneas nem barreiras diferentes consecutivas portanto não analisamos o comportamento do barramento especial nestas situações.

7. REFERÊNCIAS BIBILIOGRÁFICAS

[1] TORRES, M.X. **Soluções para sincronização de barreira em máquinas paralelas MIMD.** São Paulo, 1994. 156p. Dissertação (Mestrado) - Escola Politécnica, Universidade de São Paulo.

[2] CHEN, T.; CHUAN-QI, Z. A new synchronization mechanism. In: INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING, 20., Anaheim, 1991. **Proceedings**. CRC Press, 1991. v.1, p. 176-9.

[3] PERRON, R.; MUNDIE, C. The architecture of the Alliant FX/8. In: IEEE COMPUTER SOCIETY INTERNATIONAL CONFERENCE (COMPCON SPRING), 31., San Francisco, 1986. **Digest of Papers**. New York, IEEE, 1986. p. 390-3.

[4] TORRES, M.X. et al. Um barramento especial para sincronização de barreira em máquinas de memória compartilhada. In: SIMPÓSIO BRASILEIRO DE ARQUITETURA DE COMPUTADORES PROCESSAMENTO DE ALTO DESEMPENHO, 6., Caxambú, 1994. **Anais**. Imprensa universitária UFMG, 1994. p. 213-28.

[5] HWANG, K.; SHANG, S. Wired-NOR barrier synchronization for designing large shared-memory multiprocessors. In: INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING, 20., Anaheim, 1991. **Proceedings**. CRC Press, 1991. v.1, p. 171-5.

[6] MATSUMOTO, T. et al. MISC: a mechnism for integrated synchronization and communication using snoop caches. In: INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING, 20., Anaheim, 1991. **Proceedings**. CRC Press, 1991. v.1, p. 161-70.

[7] TORRES, M.X., et al. Estudo do efeito da sincronização de barreira implementada em software no desempenho de máquinas paralelas. In: SIMPOSIO BRASILEIRO DE ARQUITETURA DE COMPUTADORES PROCESSAMENTO DE ALTO DESEMPENHO,5. Florianópolis, 1993. **Anais**. Florianópolis, 1993. v.2, p. 243-258.

[8] VEENSTRA, J.E.; FOWLER, R.J. **MINT tutorial and user manual**. Rochester, Computer Science Departament The University of Rochester, June 1993. (Technical Report 452).

[9] SINGH, J.P., et al.,. **SPLASH: Stanford parallel applications for shared-memory**. Stanford, Computer Systems Laboratory Stanford University, Apr. 1991. (CSL-TR-91-469).