Avaliação de Desempenho de Algoritmos de Balanceamento de Carga de Aplicações SPMD na Presença de Carga Externa*

Roberto Costa† Daniela Vianna‡ Viviane Thomé§ Alexandre Plastino¶

Universidade Federal Fluminense, Departamento de Ciência da Computação
Rua Passo da Pátria, 156 - Bloco E - 3º andar - Boa Viagem - 24210-240 - Niterói - RJ
{rcosta,vthome,dvianna}@ic.uff.br, plastino@dcc.ic.uff.br

Resumo

O desempenho de programas paralelos é freqüentemente afetado por diferentes fatores dinâmicos de desequilíbrio de carga. Um fator de desequilíbrio muito comum, presente nos ambientes paralelos não dedicados, é a existência de outros processos concorrendo com a aplicação paralela pelos recursos computacionais. A heterogeneidade e variação desta carga externa impede que seja feita uma distribuição prévia e equilibrada das tarefas da aplicação paralela. O uso de uma estratégia de balanceamento de carga adequada é fundamental para a redução dos efeitos causados por este fator de desequilíbrio. Neste trabalho, serão avaliadas oito estratégias de balanceamento de carga utilizadas em duas aplicações paralelas SPMD distintas, quando executadas na presença de carga externa. As aplicações SPMD implementadas calculam: (a) a multiplicação de matrizes quadradas e (b) a dispersão térmica em meios porosos. Neste trabalho, propõe-se também um índice de desbalanceamento de carga para aplicações SPMD.

1. Introdução

Algoritmos paralelos são projetados segundo dois modelos de programação: paralelismo de função e paralelismo de dados. Este último, quando associado a arquiteturas MIMD (*Multiple Instruction, Multiple Data*), é conhecido como SPMD (*Single Program, Multiple Data*) [10]: o mesmo programa é executado, sem sincronismo no nível de instrução, nos diferentes processadores, manipulando conjuntos distintos de dados (tarefas).

O modelo SPMD tem sido amplamente utilizado devido à facilidade de desenvolvimento e controle de um único código executado nos diversos processadores. Além disso, muitos problemas podem ser resolvidos por algoritmos baseados neste modelo [5].

O desempenho de uma aplicação paralela SPMD pode ser afetado por diversos fatores que causam o desequilíbrio de carga, tais como: a heterogeneidade da arquitetura paralela adotada, o desconhecimento da quantidade de processamento envolvida em cada tarefa, a criação dinâmica e a migração de tarefas, além da heterogeneidade e da variação da carga externa à aplicação nos diversos processadores de um ambiente paralelo não dedicado. A utilização de uma estratégia adequada de balanceamento de carga é fundamental para redução do efeito desses fatores de desequilíbrio.

No caso do modelo SPMD, a estratégia de balanceamento de carga aparece freqüentemente inserida na própria aplicação, o que exige do projetista uma preocupação com a escolha da estratégia mais adequada. Devido ao grande número de algoritmos já definidos [1, 2, 3, 4, 13, 14], identificar aquele ideal à aplicação sendo desenvolvida pode não ser uma tarefa simples.

Um dos objetivos deste trabalho é avaliar o comportamento de oito estratégias de balanceamento de carga utilizadas em duas aplicações paralelas SPMD distintas, quando executadas na presença de carga externa. As aplicações SPMD implementadas calculam: (a) a multiplicação de matrizes quadradas e (b) a dispersão térmica em meios porosos (chamada aplicação dos *termions*) [6, 11].

Estas duas aplicações não apresentam fatores internos de desequilíbrio: todas as tarefas possuem aproximadamente a mesma demanda computacional e não há criação dinâmica de tarefas. Isto permite a realização de uma análise isolada do efeito da carga externa sobre a aplicação.

Estas aplicações foram desenvolvidas utilizando-se a ferramenta de programação paralela SAMBA (*Single Applica-*

*Este trabalho foi desenvolvido com o apoio do CNPq e da FAPERJ.
†Bolsista de Iniciação Científica FAPERJ (152891/2000).
‡Bolsista de Iniciação Científica CNPq (181611/01-5).
§Bolsista de Iniciação Científica CNPq (181612/01-1).
¶Bolsista do Programa CNPq Kit-Enxoval (680123/01-6) e do Programa FAPERJ APQ1 (171352/2000).

tion, Multiple Load Balancing) [7, 8, 9]. SAMBA é um *framework* que captura a estrutura e as características comuns a diferentes aplicações SPMD, facilitando o desenvolvimento destas.

As estratégias de balanceamento de carga avaliadas pertencem à biblioteca de balanceamento de carga da ferramenta SAMBA. Neste trabalho, pretende-se também avaliar o comportamento específico de uma nova versão, implementada pelos autores deste trabalho e incluída na biblioteca, de uma das estratégias disponíveis.

Neste trabalho, propõe-se também um índice de desbalanceamento de carga para aplicações SPMD. Este índice mede o nível de desbalanceamento de uma execução de uma aplicação paralela SPMD.

Na próxima seção, descreve-se a biblioteca de balanceamento de carga da ferramenta SAMBA, com a nova estratégia incluída. Na Seção 3, o índice de desbalanceamento de carga é definido. Na Seção 4, são apresentados os resultados computacionais e a avaliação de desempenho dos algoritmos de balanceamento de carga quando utilizados com a multiplicação de matrizes. Na Seção 5, as estratégias que apresentaram os melhores desempenhos na avaliação anterior são novamente avaliadas com a aplicação dos *termions*. As conclusões finais são apresentadas na Seção 6.

2. Biblioteca de algoritmos de balanceamento de carga

Uma das funções da ferramenta SAMBA é gerar aplicações SPMD a partir, basicamente, da codificação das rotinas de geração e execução das tarefas da aplicação. Na primeira, empacotam-se os dados que representam as tarefas. A segunda representa o código comum que será executado sobre todos os dados (tarefas). Uma vez gerada a aplicação, o usuário pode adotar qualquer um dos algoritmos da biblioteca de balanceamento de carga da ferramenta, sem custo algum de reprogramação.

O projeto desta biblioteca foi guiado pela taxonomia de algoritmos de balanceamento de carga apresentada em [7]. Os algoritmos da biblioteca implementam diferentes classes e características tratadas nesta taxonomia. Os algoritmos utilizados são descritos a seguir. O algoritmo inserido na biblioteca para ser avaliado neste trabalho é o sob demanda 2 (BC3).

Algoritmo BC1 - Estático - O processador mestre, que contém o conjunto inicial de tarefas, as distribui igualmente entre todos os processadores, incluindo ele próprio. Em seguida, executam-se as tarefas sem que nenhuma ação visando o balanceamento dinâmico de carga seja considerada.

Algoritmo BC2 - Sob Demanda 1 - Neste algoritmo, inicialmente, o processador mestre distribui um bloco de tarefas para cada processador escravo. O número de tarefas em cada bloco é definido por um parâmetro de ativação da aplicação. Após o recebimento do bloco inicial, cada processador escravo pede um novo bloco ao mestre quando encerra a execução das tarefas do bloco anterior. Enquanto houver blocos de tarefas, o processador mestre atende aos pedidos de novos blocos, enviando-os aos processadores solicitantes. O processador mestre não executa tarefas.

Algoritmo BC3 - Sob Demanda 2 - Este algoritmo tem como base o anterior, exceto o fato de que o processador mestre também executa tarefas da aplicação. Sempre que o mestre encontra-se ocioso, sem requisições para atender, executa uma tarefa da aplicação. Após a execução desta tarefa, o mestre verifica se algum processador escravo requisitou um novo bloco de tarefas. Após atender todos os pedidos, o mestre volta a executar mais tarefas. Este procedimento é executado enquanto houver tarefas.

Algoritmo BC4 - Distribuído, Síncrono, Global e Coletivo - Neste algoritmo, o processador mestre, que detém o conjunto inicial de tarefas, as distribui igualmente entre todos os processadores, incluindo ele próprio. Em seguida, sempre que um processador encerra as sua tarefas, envia uma mensagem aos outros processadores solicitando a execução síncrona do balanceamento de carga. Todos os processadores, então, enviam aos demais o seu índice de carga interna (a quantidade de tarefas ainda não executadas). Em seguida, todos os processadores definem as transferências necessárias ao equilíbrio exato da carga em todos os processadores. Finalmente, cada processador executa as transferências nas quais está envolvido.

Algoritmo BC5 - Centralizado, Global e Coletivo - Este algoritmo é uma versão centralizada do anterior. A definição das transferências necessárias ao equilíbrio exato é realizada por um processador central e não por todos os processadores. Sempre que um processador encerra suas tarefas, envia uma mensagem aos outros processadores solicitando que estes enviem ao processador mestre o índice de carga interna, no caso, a quantidade de tarefas ainda não executadas. O mestre define, então, as transferências necessárias ao equilíbrio exato da carga em todos os processadores e informa aos escravos as transferências nas quais estes devem participar. Em seguida, cada processador executa as transferências indicadas.

Algoritmo BC6 - Distribuído, Assíncrono, Global e Individual - Como no Algoritmo BC4, o processador mestre distribui o conjunto inicial de tarefas igualmente entre todos os processadores, incluindo ele próprio. Em seguida, sempre que um processador encerra suas tarefas, ele envia uma mensagem aos demais solicitando o índice de carga interna de cada um. Após receber os índices, o processador que ativou o balanceamento identifica que processador possui a maior carga e o envia uma mensagem pedindo metade de sua carga. Esta transferência é então realizada.

Algoritmo BC7 - Distribuído, Síncrono, Local, Particionado, e Coletivo - Este algoritmo é uma versão local-partici-

onada do Algoritmo BC4. Aqui, o balanceamento de carga ocorre isoladamente dentro de cada grupo. A divulgação de índices de carga, assim como as transferências de tarefas, ocorre somente entre processadores de um mesmo grupo. Cada processador pertence somente a um grupo. Os grupos são definidos pelo usuário em um arquivo, cujo nome é informado como parâmetro de ativação da aplicação.

Algoritmo BC8 - Distribuído, Assíncrono, Local, Por Vizinhança e Individual - Este algoritmo é uma versão local-por-vizinhança do Algoritmo BC6. Aqui, a divulgação de índices de carga, assim como as transferências de tarefas, ocorre somente entre processadores de um mesmo grupo. Os grupos são definidos considerando-se a vizinhança lógica entre os processadores. Neste caso, cada processador pode pertencer a mais de um grupo. A vizinhança lógica é especificada pelo usuário em um arquivo, cujo nome é informado como parâmetro de ativação da aplicação.

3. Índice de desbalanceamento de carga

Para medir o desbalanceamento de carga em uma execução de uma aplicação paralela, neste trabalho é proposto um *índice de desbalanceamento de carga* (IDC). O IDC de uma execução de uma aplicação paralela, em p processadores, é definido por $IDC = TMO/t_f$, como sendo o *tempo médio de ócio dos processadores* (TMO) em relação ao tempo gasto pelo processador que por último concluiu suas tarefas (t_f). O TMO é definido a seguir.

$$TMO = \frac{\sum_{i=1}^{p}(t_f - t_i)}{(p-1)} \quad (1)$$

onde t_i é o tempo gasto pelo *i-ésimo* processador ($1 \leq i \leq p$). No máximo $p-1$ processadores ficam ociosos (esperando que o último encerre suas tarefas) e contribuem para o cálculo de TMO. São consideradas p subtrações (e não $p-1$) no somatório pois pelo menos uma delas será necessariamente zero.

Para ilustrar a definição deste índice, considere a Figura 1 com três execuções de uma aplicação paralela fictícia em 4 processadores, com níveis de desbalanceamento de carga distintos. Cada barra dos gráficos representa o tempo (em segundos) de execução em cada processador.

Na execução E1, o tempo médio de ócio é 2s (dado por $\frac{(10-8)+(10-9)+(10-7)}{3}$). O índice de desbalanceamento desta execução é 20%, ou seja o quanto 2s representa dentro do tempo que o processador mais lento levou para realizar suas tarefas (10s). A execução E2 representa o equilíbrio total e, portanto, seu índice de desbalanceamento é 0%. A execução E3 possui o desbalanceamento máximo (100%) – apenas um processador executou todas as tarefas.

Cabe ressaltar que este índice não visa medir a eficiência de um algoritmo de balanceamento de carga. Pois um determinado algoritmo pode gerar uma execução 100% balanceada, porém o custo de sua execução pode comprometer o desempenho da aplicação em questão.

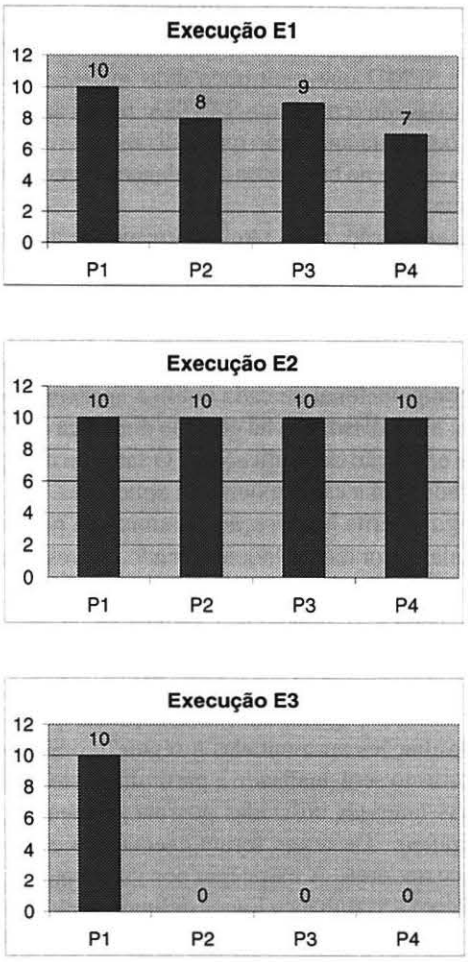

Figura 1. Ilustração de três execuções com desbalanceamentos distintos

4. Avaliação de desempenho - multiplicação de matrizes

Os experimentos computacionais reportados neste trabalho foram realizados em um *cluster* composto por 32 processadores IBM PC Pentium II 400 MHz, conectados por um *switch* IBM 10 Mbits. Cada máquina possui 32 MB de RAM, exceto uma, que possui 64 MB.

Cada experimento foi realizado apenas uma vez. A repetição de qualquer experimento levava ao mesmo resultado uma vez que o ambiente se encontrava em modo exclusivo e as estratégias de balanceamento de carga não apresentavam

fatores aleatórios a serem considerados.

Nesta seção, apresenta-se a avaliação de desempenho da aplicação SPMD que multiplica duas matrizes quadradas. Com o auxílio da ferramenta SAMBA, foram desenvolvidas oito versões desta aplicação que utilizam diferentes algoritmos disponíveis na biblioteca de balanceamento de carga da ferramenta.

Nesta aplicação, cada tarefa corresponde ao cálculo de uma linha da matriz resultante, ou seja, corresponde à multiplicação de uma linha da primeira matriz por todas as colunas da segunda. Considerando-se duas matrizes com dimensão nxn, haverá n tarefas a serem executadas. A demanda computacional de cada tarefa é aproximadamente a mesma e, além disso, não há geração dinâmica de tarefas ao longo da execução das aplicações. O fator de desequilíbrio considerado será a carga externa às aplicações.

A carga externa à aplicação foi simulada por processos que calculam, por tempo indeterminado, expressões aritméticas. Estes processos são disparados e controlados em um ambiente dedicado, sem a concorrência de outros usuários. Em cada grupo de quatro processadores, um não tem carga externa, outro executa um processo de carga, outro executa dois processos e o quarto executa três processos.

Nas avaliações apresentadas a seguir, o desempenho de cada algoritmo será analisado a partir do seu tempo de execução. As matrizes utilizadas possuem dimensão n=1500 (1500 tarefas). Os testes foram executados com 4, 8, 16 e 32 processadores. A estratégia por vizinhança (BC8) foi avaliada com a vizinhança lógica definida como um anel. A Figura 2 ilustra graficamente os tempos de execução (em segundos) de cada estratégia de balanceamento de carga apresentada na Seção 2.

A execução da estratégia estática (BC1) evidencia o efeito da carga simulada. Na execução com 4 processadores, cada um realizou 375 tarefas. Os tempos apresentados pelos processadores foram: 70s, 128s, 191s e 254s. Utilizando-se a Fórmula 1 (Seção 3), tem-se um índice de desbalanceamento igual a 49%. Nas execuções com 8, 16 e 32 processadores os índices de desbalanceamento foram, respectivamente, 43%, 39% e 39%. Estes valores mostram a necessidade de se adotar um algoritmo dinâmico de balanceamento de carga.

Considerando-se 4 processadores, o tempo da estratégia sob demanda 1 (BC2) é muito maior do que o tempo das demais estratégias dinâmicas (BC3-8), devido ao fato de apenas três dos quatro processadores executarem tarefas. O quarto processador, nesta estratégia, fica dedicado à distribuição de tarefas. Com poucos processadores, esta perda representa um custo alto para a estratégia. Esta perda foi eliminada com a implementação da nova estratégia sob demanda (BC3), em que o mestre também executa tarefas. Esta estratégia obteve o melhor desempenho com quatro processadores. As estratégias por transferência (BC4-8) apre-

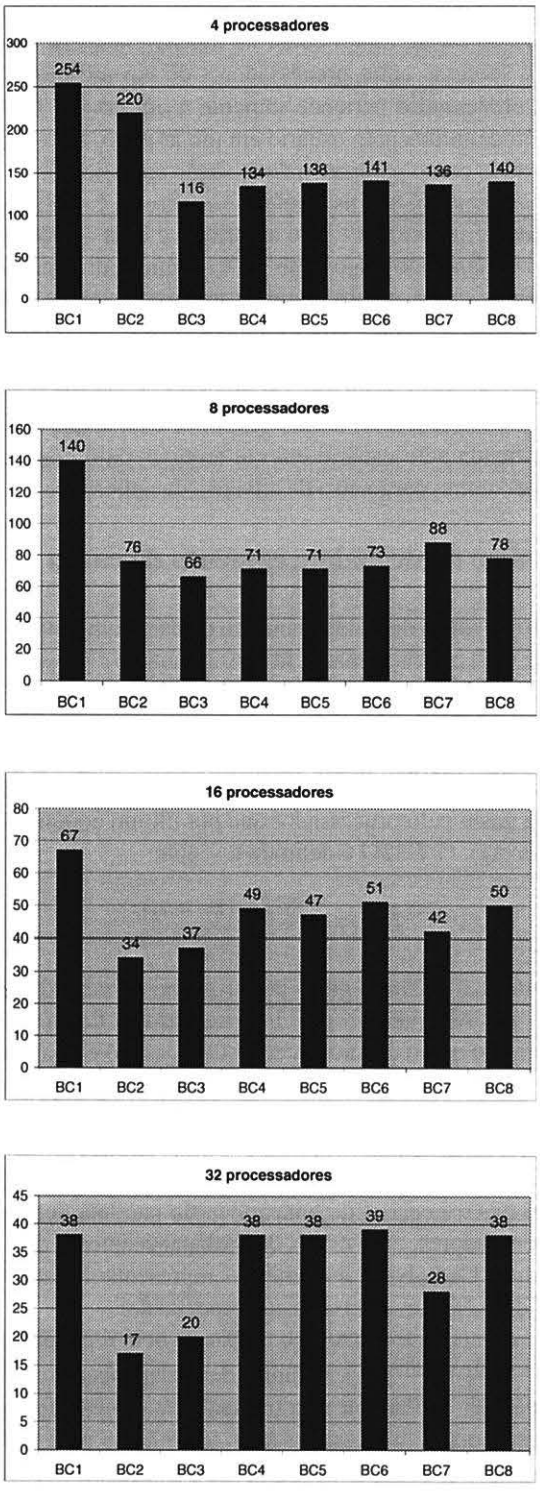

Figura 2. Tempos dos diferentes algoritmos de balanceamento de carga para a multiplicação de matrizes

sentaram desempenhos semelhantes entre si com 4 processadores. Neste caso, a diferença entre a média dos tempos das estratégias por transferência (BC4-8) e a nova estratégia sob demanda (BC3) representa aproximadamente 16% da referida média.

Com 8 processadores, o algoritmo BC2 continua apresentando um desempenho inferior ao do BC3. Este último obteve novamente o melhor desempenho. Porém, a diferença entre eles é bem menor do que na execução com 4 processadores. O fato de o BC2 alocar um processador exclusivo para a distribuição de tarefas compromete menos a estratégia com o aumento do número de processadores. Isto já se evidencia com 8 processadores e se confirma com 16 e 32 processadores. Nestas últimas, o desempenho do BC2 supera o do BC3. Verifica-se então que o BC2 se beneficia do aumento do número de processadores. Porém, o BC3, ao contrário, tem seu desempenho reduzido, pois a execução de tarefas no mestre pode comprometer o pronto atendimento das requisições de um número maior de processadores. Vale ressaltar que, de modo geral, o desempenho do BC3 foi melhor do que o do BC2, pois o BC3 quando não foi o melhor algoritmo foi o segundo melhor. Enquanto que o BC2 foi o melhor com 16 e 32 processadores, porém com 4 processadores só não foi pior do que a estratégia estática (BC1).

As estratégias por transferência (BC4-8) apresentaram sempre desempenho inferior a pelo menos uma das estratégias sob demanda (BC2,3). As estratégias BC4-8 apresentaram desempenhos semelhantes entre si com 4 e 8 processadores. Porém, com 16 e 32, principalmente neste último caso, observa-se que o algoritmo particionado (BC7) teve um melhor desempenho que sua versão global (BC4). Com o aumento do número de processadores, ficam mais caras, na estratégia global (BC4), a sincronização e a comunicação entre todos os processadores para troca de índices de carga. Isto evidencia a necessidade de se particionar o conjunto total de processadores com o seu crescimento.

Todas as execuções das estratégias sob demanda foram realizadas com blocagem 1 (uma tarefa por bloco). A Tabela 1 evidencia que o aumento do número de tarefas por bloco, até um certo nível, melhora levemente o desempenho da estratégia sob demanda 2 (BC3). A execução de BC3 em 8 processadores com blocagem 4 apresentou um desempenho melhor (61s) do que com blocagem 1 (66s). Porém um aumento significativo do número de tarefas por blocos pode comprometer seu desempenho. A execução de BC3 em 16 processadores com 32 tarefas por bloco apresentou um desempenho pior (43s) do que com uma tarefa por bloco (37s). O aumento da blocagem diminui o número de requisições ao mestre e o libera para executar mais tarefas. Porém, seu aumento excessivo pode levar um processador com carga externa alta a pegar o último bloco e prolongar o fim da aplicação.

Tabela 1. Tempos em segundos da estratégia BC3 variando a blocagem

	Qtd. de tarefas por bloco (blocagem)					
Processadores	1	2	4	8	16	32
4	116	115	115	115	121	123
8	66	63	61	61	62	62
16	37	34	33	33	33	43
32	20	18	18	19	24	26

5. Avaliação de desempenho - aplicação dos termions

Nesta seção, serão avaliadas as estratégias que obtiveram os melhores desempenhos na avaliação anterior (as estratégias sob demanda), quando utilizadas com a aplicação dos *termions*. Esta aplicação calcula a dispersão térmica em meios porosos [6, 11]. O meio poroso considerado é composto por elementos sólidos e por fluido. A dispersão térmica é avaliada pelo movimento de partículas (chamadas *termions*) identificadas por um par de coordenadas (x, y), que indicam sua posição e, indiretamente, o meio no qual elas se encontram.

Inicialmente, um grande número de *termions* é liberado. A posição das partículas é alterada a cada iteração (passo) de acordo com uma nova direção (determinada aleatoriamente) e com o tamanho do passo, que depende das propriedades térmicas do meio no qual a partícula se encontra e da velocidade do fluxo do fluído (caso a partícula se encontre na parte fluida). As partículas alcançam suas posições finais após um determinado número de passos, obtendo-se desta distribuição a dispersão térmica.

Cada *térmion* percorre um caminho aleatório. Caso o meio tenha propriedades térmicas heterogêneas, a avaliação da trajetória destas partículas pode não exigir a mesma demanda computacional. Neste trabalho, o meio possui propriedades térmicas homogêneas, garantindo a inexistência de fatores internos de desequilíbrio de carga. Esta escolha foi motivada pelo interesse em se avaliar o fator de desequilíbrio externo, provocado pela carga externa. Em [12], foram utilizadas propriedades térmicas diferentes com o objetivo de avaliar o desbalanceamento causado por esta característica.

Foram geradas três versões desta aplicação: a primeira utilizando o algoritmo estático e as demais utilizando as estratégias sob demanada BC2 e BC3. Nesta avaliação, foram utilizados 1000 *termions* (1000 tarefas). Cada tarefa consiste em calcular o percurso de um *térmion*. Cada tarefa desta aplicação demanda um custo computacional maior do que o custo das tarefas da multiplicação de matrizes. As cargas artificiais foram distribuídas seguindo o mesmo padrão da

avaliação anterior, exceto pelo fato de terem sido alocados três processos de carga no processador mestre. Desta forma será avaliado o comportamento das estratégias sob demanda com o processador mestre sobrecarregado.

A Figura 3 ilustra os tempos de execução (em segundos) das estratégias sob demanda e estática para a aplicação dos *termions*. O índice de desbalanceamento para a estratégia estática (BC1) com 4, 8, 16 e 32 processadores são, respectivamente: 25%, 32%, 36% e 39%.

Verifica-se que o comportamento apresentado pelas estratégias sob demanda é similar aos resultados obtidos na aplicação da multiplicação de matrizes. Com 4 processadores o desempenho da estratégia sob demanda 2 (BC3), continua sendo superior ao da estratégia BC2, mesmo com o processador mestre sobrecarregado, o que demonstra a importância de um processador a mais computando tarefas quando há poucos processadores. Na execução com 8 processadores, o desempenho da estratégia BC2 passou a ser melhor do que o da estratégia BC3. Na aplicação anterior, essa troca se deu apenas com 16 e 32 processadores. Na execução da aplicação dos *termions*, o mestre encontrava-se sobrecarregado (com três cargas artificiais) e a atribuição de tarefas ao mestre, além deste ter que atender as requisições dos escravos, parece comprometer o desempenho da estratégia BC3.

Com 16 e 32 processadores, o BC2 permanece superior ao BC3, o que confirma que, com o aumento do número de processadores, a execução de tarefas no mestre compromete o pronto atendimento das requisições dos processadores escravos.

6. Conclusões

Este trabalho apresentou a avaliação de desempenho de oito estratégias de balanceamento de carga quando utilizadas em duas aplicações paralelas SPMD. A estratégia sob demanda BC3 implementada e incluída na biblioteca da ferramenta SAMBA apresentou um ótimo desempenho. Quando não foi a melhor estratégia, obteve desempenho inferior apenas ao da estratégia sob demanda original BC2. Este resultado sugere a implementação de uma estratégia híbrida sob demanda que, de acordo com o número de processadores disponível, decida se o mestre deve (BC3) ou não (BC2) também executar tarefas. Uma outra possibilidade de investigação é a implementação de uma variação da estratégia sob demanda BC3 na qual são executados dois processos no processador mestre: um para atender os pedidos dos escravos e outro para executar tarefas da aplicação.

Outras sugestões para trabalhos futuros apontam para uma avaliação destes algoritmos utilizando-se uma simulação de carga externa mais próxima da situação real – na qual o número de processos em cada processador varia ao longo da execução da aplicação. Além disso, um estudo sobre a

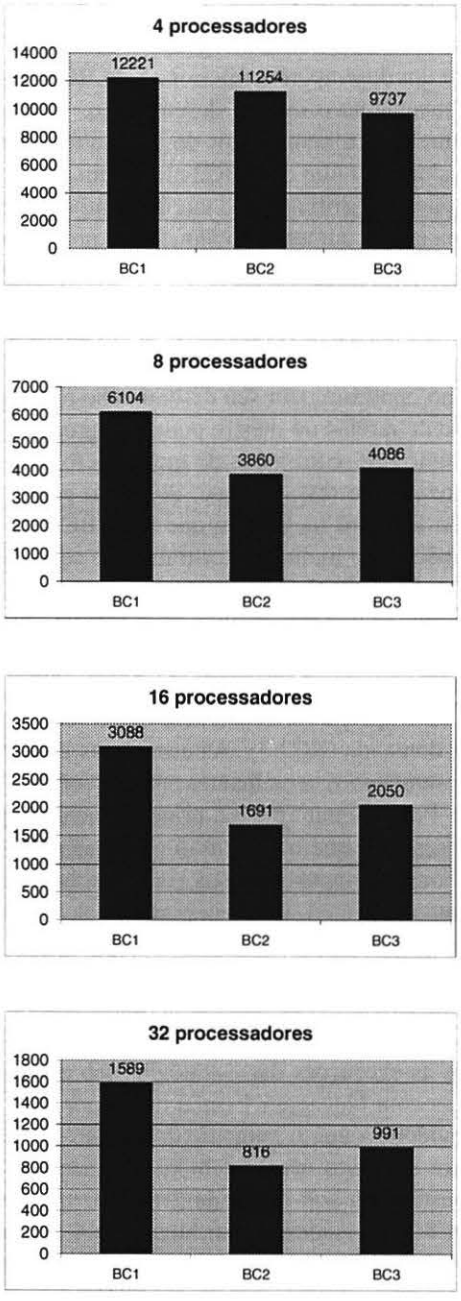

Figura 3. Tempos dos diferentes algoritmos de balanceamento de carga para a aplicação dos termions

utilização de um índice de carga externa, como sugerido em [7], pode levar à definição de uma estratégia eficiente para aplicações SPMD executadas na presença de carga externa.

Referências

[1] J.N.C. Árabe e C.D. Murta, "Auto-Balanceamento de Carga em Programas Paralelos", *Proceedings of the VIII Brazilian Symposium on Computer Architecture and High Performance Processing*, 1996, 161–171.

[2] M.A. Franklin e V. Govindan, "A General Matrix Iterative Model for Dynamic Load Balancing", *Parallel Computing* 22 (1996), 969–989.

[3] M. Furuichi, K. Taki e N. Ichiyoshi, "A Multi-Level Load Balancing Scheme for Or-Parallel Exhaustive Search Programs on the Multi-Psi", *Proceedings of the Second ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 1990, 50–59.

[4] S. Lifschitz, A. Plastino e C.C. Ribeiro, "Exploring Load Balancing in Parallel Processing of Recursive Queries", *Proceedings of the III Euro-Par Conference, Lecture Notes in Computer Science* 1300, 1997, 1125–1129.

[5] T. G. Mattson, "Scientific Computation", *Parallel and Distributed Computing Handbook* (A.Y. Zomaya, editor), 981–1002, McGraw-Hill, 1996.

[6] C. Moyne, S. Didierjean, H.P.A. Souto e O.T. da Silveira Filho, "Thermal Dispersion in Porous Media: One-Equation Model", *International Journal of Heat and Mass Transfer* 43, 2000, 3853–3867.

[7] A. Plastino, *Balanceamento de Carga de Aplicações Paralelas SPMD*, Tese de Doutorado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, 2000.

[8] A. Plastino, C.C. Ribeiro e N. Rodriguez, "A Tool for SPMD Application Development with Support for Load Balancing", *Proceedings of the International Conference ParCo'99*, 639–646, Imperial College Press, 2000.

[9] A. Plastino, C.C. Ribeiro e N. Rodriguez, "A Framework for SPMD Applications with Load Balancing", *Proceedings of the 12th Symposium on Computer Architecture and High Performance Computing* (SBAC-PAD'2000), 245-252, São Pedro, 2000.

[10] M.J. Quinn, *Parallel Computing: Theory and Practice*, Series in Computer Science, McGraw-Hill, 1994.

[11] O.T. da Silveira Filho, *Dispersão Térmica em Meios Porosos Periódicos. Um Estudo Numérico.*, Tese de Doutorado, Instituto Politécnico, Universidade Estadual do Rio de Janeiro, 2001.

[12] V. Thomé, D. Vianna, R. Costa, A. Plastino e O.T. da Silveira Filho, "Exploring Load Balancing in a Scientific SPMD Parallel Application", *Proceedings of the 4th International Workshop on High Performance Scientific and Engineering Computing with Applications* (HPSECA'2002) realizado em conjunto com *31st International Conference on Parallel Processing* (ICPP'2002), a ser publicado.

[13] M.A. Willebeek-LeMair e A.P. Reeves, "Strategies for Dynamic Load Balancing on Highly Parallel Computers", *IEEE Transactions on Parallel and Distributed Systems* 4 (1993), 979–993.

[14] M.J. Zaki, W. Li e S. Parthasarathy, "Customized Dynamic Load Balancing for a Network of Workstations", *Journal of Parallel and Distributed Computing* 43 (1997), 156–162.