Auto-Balanceamento de Carga em Programas Paralelos

José Nagib Cotrim Árabe *
Departamento de Ciência da Computação
Universidade Federal de Minas Gerais

Cristina Duarte Murta †
Departamento de Informática
Universidade Federal do Paraná

Resumo

Este artigo aborda o problema do balanceamento interno de carga de programas paralelos em redes de *workstations* homogêneas e heterogêneas. O conceito de programas paralelos auto-balanceáveis é discutido. Um estudo detalhado do balanceamento de carga de programas Dome [1] é apresentado. Os principais objetivos são quantificar os ganhos obtidos com o balanceamento da carga e identificar os obstáculos mais importantes para a obtenção de ganhos adicionais. Ambientes homogêneos e heterogêneos, estáveis e instáveis foram analisados. Os resultados indicam que a obtenção de ganhos relevantes com o balanceamento de carga está vinculada a uma escolha cuidadosa do número de operações paralelas que devem ser executadas antes da próxima etapa de balanceamento de carga e que este número deve ser dinamicamente determinado por algum algoritmo adaptativo em função da heterogeneidade e da estabilidade do ambiente.

Abstract

This paper addresses the problem of internal load balancing for parallel programs on networks of homogeneous and heterogeneous workstations. The concept of self-balancing in parallel programming is discussed. A detailed study of the load balancing of Dome [1] programs is presented. The goals are to quantify the performance gains and to identify the hurdles that prevent additional gains. Homogeneous and heterogeneous, stable and unstable environments are investigated. The results indicate that getting additional gains in performance depends on a careful choice of the number of parallel operations that must be executed before the next load balancing phase. Furthermore, this choice must be done dynamically by some adaptive algorithm according to the environment stability and heterogeneity.

1 Introdução

Redes heterogêneas de computadores formam um ambiente bastante atraente para processamento paralelo. As principais vantagens são a disponibilidade deste tipo de

*E-mail:arabe@dcc.ufmg.br. Endereço: caixa postal 702, cep 31270-010, Belo Horizonte, MG.
†Doutoranda no DCC/UFMG, com bolsa da CAPES. E-mail: cristina@dcc.ufmg.br.

ambiente, a possibilidade de obtenção de alto desempenho devido à utilização das múltiplas CPUs para computação paralela e a extensibilidade a baixo custo.

Assumindo paralelismo suficiente do programa, o uso efetivo deste potencial encontra obstáculos importantes dos quais podemos citar o gerenciamento dos recursos disponíveis neste ambiente, a comunicação eficiente de hardware e software e algumas ineficiências inerentes ao processo de paralelização, tais como sincronizações exigidas pelo programa, gerência do paralelismo e computação redundante. O objetivo da gerência de recursos é maximizar o desempenho geral do sistema. Para isso é necessário o balanceamento de carga, isto é, a adequada distribuição do trabalho entre os processadores.

Os modelos tradicionais de balanceamento externo de carga abordam o problema de gerenciamento de carga de processos sequenciais distribuídos e assumem a existência de uma entidade que tem a visão global do conjunto de recursos e que toma decisões de alocação em função desta informação [2, 3, 6, 8]. Neste modelo, uma vez alocado, em geral o processo não é interrompido para realocação.

Uma nova abordagem para balanceamento interno de carga de programas paralelos foi introduzida pelo sistema Dome (*Distributed object migration environment*) [1]. Programas em Dome são constituídos por uma coleção de objetos paralelos instanciados a partir das classes que o sistema oferece, implementadas como uma biblioteca em C++. O paradigma de programação paralela é o SPMD (*single program multiple data*). A criação da máquina paralela virtual, o controle de processos e a comunicação são implementados com a utilização do ambiente PVM [4].

No ambiente Dome, os objetos que constituem os dados do programa são distribuídos entre os processadores. Cada processador executa um número de operações sobre os objetos alocados a ele e, a seguir, os processadores trocam informação entre si para verificação do desempenho relativo. Se necessário, há uma redistribuição dos objetos entre os processadores. Isto significa que o programa faz um auto-balanceamento, redistribuindo seus próprios dados.

O objetivo deste trabalho é analisar o desempenho de programas paralelos auto-balanceáveis, apontando as vantagens, os problemas e algumas soluções para esta abordagem. O trabalho foi realizado tendo como base o sistema Dome. Foi utilizada a simulação com técnica de avaliação do desempenho. A partir dos resultados aqui obtidos algumas alternativas serão implementadas e avaliadas no sistema real, o que deverá validar as simulações realizadas.

Foram feitos experimentos para ambientes estáveis e instáveis, homogêneos e heterogêneos. Os resultados indicam que o parâmetro mais crítico é a determinação do momento em que o balanceamento de carga deve ser feito, isto é, qual é a quantidade de trabalho que deve ser executada antes da próxima fase de balanceamento de carga. Esta escolha é crítica tanto em ambientes estáveis quanto instáveis e deve ser feita por políticas adaptativas considerando a heterogeneidade e a estabilidade do ambiente. Vários outros aspectos importantes no contexto de programas paralelos auto-balanceáveis foram analisados e os resultados são apresentados neste artigo.

Este artigo contém 6 seções das quais esta é a primeira. A segunda seção descreve a estratégia de balanceamento interno da carga de programas paralelos Dome bem como os parâmetros relevantes. A seção seguinte apresenta o modelo de custo de utilização dos recursos processador e rede no balanceamento. A quarta seção apresenta os resultados e finalmente a última conclui o trabalho e propõe alguns caminhos para sua continuação.

2 Balanceamento de Carga de Programas Paralelos Dome

O sistema Dome é um ambiente para especificação e execução de programas paralelos em redes homogêneas e heterogêneas. Este sistema tem como objetivos facilitar a programação e otimizar a execução de aplicações paralelas. Os programas são automaticamente distribuídos na rede. O sistema provê balanceamento dinâmico da carga e tolerância a falhas.

Os programas Dome seguem o modelo de programação paralela SPMD e são constituídos por uma coleção de objetos Dome. Cada objeto, por sua vez, é constituído por um arranjo linear de elementos. O tempo de execução de um programa paralelo Dome num ambiente multiusuário heterogêneo sofre influências de desbalanceamentos internos e externos. O desbalanceamento externo tem origem nas diferentes capacidades das máquinas, nas cargas específicas de cada nodo durante a execução e na heterogeneidade da própria rede. A solução deste problema é a redistribuição do trabalho gerado no sistema (na forma de tarefas independentes) entre os processadores. Desbalanceamento interno surge devido ao próprio contexto do programa paralelo em execução — a distribuição do trabalho entre tarefas paralelas pode mudar. Neste caso é feita a redistribuição das tarefas paralelas entre os processadores. O sistema Dome trata do balanceamento da carga interna de um programa paralelo levando em consideração os desbalanceamentos externos, refletidos na latência de execução das próprias operações do programa.

No início da execução, um programa Dome distribui igualmente os elementos de cada um de seus objetos entre os processadores. A seguir cada processador executa um número pré-determinado de operações Dome sobre os elementos. Uma operação Dome é uma operação executada sobre um objeto Dome, isto é, sobre todos os elementos do objeto. O tempo necessário para executar as operações em cada processador é medido. Este valor, utilizado como índice de carga, é uma medida da capacidade de processamento da máquina no período de medição e é função principalmente da velocidade do processador e da carga. Portanto, o índice de carga utilizado como critério de balanceamento é a taxa de execução do próprio programa Dome em cada máquina. Esta abordagem de uso da própria aplicação como *benchmark* fornece resultados mais precisos do que os índices de carga normalmente utilizados [3], por exemplo, tamanho da fila de processos, além de contribuir para o término da execução da tarefa.

Observe que para haver a comparação entre os tempos de execução de cada máquina é necessária uma sincronização após o processamento das operações. Antes da primeira fase de balanceamento, a execução de um número pequeno de operações é suficiente para que se obtenha a informação necessária para redistribuição da carga. Após a comparação dos tempos, os objetos são redistribuídos de modo que o trabalho das próximas operações seja completado na menor quantidade possível de tempo.

O balanceamento de carga no Dome pode ser feito de forma local ou global, à escolha do usuário. Cada fase de balanceamento é composta de duas etapas, controle e movimentação de dados. Na etapa de controle, a taxa de execução observada em cada processador é comparada com as demais. Na segunda etapa os elementos são redistribuídos entre os processadores. Na etapa de controle do balanceamento

global, uma máquina é designada mestre e as demais são os escravos. Inicialmente todos os escravos enviam mensagens para o mestre contendo informação sobre o número de elementos que foram processados e o tempo gasto por elemento. A seguir o mestre calcula a distribuição ideal e envia mensagens aos escravos indicando quantos elementos cada processador deve enviar ou receber dos vizinhos. Neste tipo de balanceamento apenas a fase de controle envolve comunicação global. Como o mestre tem a visão global, é possível obter a redistribuição ideal, que considera a carga relativa entre os nodos do sistema, em uma única fase de balanceamento, movimentando dados apenas entre vizinhos.

No balanceamento local cada máquina troca mensagens apenas com seus vizinhos. Esta opção não resulta em um remapeamento de dados global ótimo após cada fase. Num ambiente estável são necessárias algumas fases de balanceamento para que seja alcançada a distribuição ótima. Quando isto ocorre dizemos que o sistema convergiu. No balanceamento global o sistema converge em uma fase.

Terminada a fase de controle, segue a movimentação dos dados, se necessário. A fase de movimentação é igual para os dois tipos de balanceamento. Para fazer a movimentação o Dome considera que as máquinas participantes estão virtualmente interconectadas em uma topologia anel ou linear. O uso destas topologias simples facilita grandemente o gerenciamento dos objetos distribuídos entre as máquinas.

3 O Modelo de Custo

Nesta seção vamos inicialmente descrever formalmente os parâmetros do sistema e demais conceitos envolvidos. A seguir vamos mostrar como é calculado o custo total de execução.

O custo total de execução é o tempo decorrido entre o início e o término da execução do programa paralelo. O objetivo final do balanceamento é minimizar este custo. Os tempos de entrada e saída, isto é, o tempo gasto na distribuição inicial dos dados entre os processadores e na exibição da resposta, não estão incluídos. Os parâmetros de um programa são o número de processadores, o número total de elementos (dimensão global dos objetos), o número total de operações Dome, o custo de uma operação básica, o número de operações antes da primeira fase de balanceamento de carga (lbinit), o número de operações entre cada fase de balanceamento (lbmod), o tipo de balanceamento (local ou global) e a topologia assumida da rede (anel ou linear). Para cada um destes parâmetros foi assumido um valor *default*, apresentado na figura 1, que foi utilizado nos experimentos descritos quando o próprio parâmetro não era uma variável.

O tempo total de execução é a soma do tempo gasto no processamento das operações do programa mais o tempo gasto no balanceamento de carga. O custo de cada fase de processamento é obtido simplesmente selecionando o maior tempo entre todos os processadores e o custo total de processamento é a soma dos maiores tempos para todas as fases de processamento. O custo total de balanceamento é a soma dos custos de controle e movimentação. Estes, por sua vez, são modelados como custos de comunicação porque envolvem basicamente troca de mensagens. Dado que acessos remotos são caros, o volume total de comunicação deve ser minimizado. O custo de cada mensagem é modelado como um custo fixo por mensagem, denominado α, mais um custo fixo por byte enviado, denominado β. α modela o *overhead*, que é

Parâmetros	Valor
número de processadores	12
dimensão global	100000
número de operações	50000
lbinit	1
lbmod	500
balanceamento	local
topologia da rede	anel

Figura 1: Valores *default* para os parâmetros

o tempo gasto pelo processador para empacotar e enviar a mensagem e também para fazer o trabalho reverso (*pack + send + receive + unpack*). α é função do tipo da rede, uma vez que a construção dos pacotes obedece a diferentes procedimentos para diferentes tipos de rede, além de ser função da velocidade do processador. β modela o *bandwidth* da rede. Os valores de α e β para rede Ethernet foram obtidos a partir da tabela 7.39 de [5] e são, respectivamente, 504 μs/mensagem e 0.91 μs/byte.

Considerando aplicações científicas, uma operação típica foi definida como uma multiplicação de ponto flutuante com custo de 10 ciclos de clock. Na implementação do simulador, os parâmetros α e β tiveram seus valores transformados para a unidade número de operações básicas. Por exemplo, para uma máquina de 100 MHz, construir uma mensagem Ethernet custa 5040 operações básicas e enviá-la custa 9 operações básicas por byte. A heterogeneidade do ambiente de simulação foi modelada fracionando-se o valor de α. Foram considerados nos experimentos 4 tipos diferentes de máquinas em termos da velocidade dos processadores. As velocidades foram expressas de forma relativa utilizando os valores {1,2,3,4}. Isto significa que uma máquina com velocidade 2 é duas vezes mais rápida do que uma com velocidade 1 e duas vezes mais lenta do que uma com velocidade 4. Uma máquina com velocidade 1 equivale à máquina de 100 MHz descrita acima. A carga externa do ambiente foi gerada aleatoriamente, com valores relativos entre 1 e 3 para cada tipo de processador. A multiplicação do índice de velocidade pelo índice de carga originou um valor de {1,..,12} que é uma medida relativa da heterogeneidade do nodo no conjunto. Acreditamos que essas suposições são razoáveis para representar redes locais de *workstations* atuais típicas. No entanto, elas não representam a presença de máquinas como supercomputadores ou computadores pessoais.

Definidos os parâmetros, podemos expressar os custos de balanceamento. A fase de controle do balanceamento global tem custo igual a

$$FC_G = \max_{i=1}^{n-1} \alpha_i + \sum_{i=1}^{n-1} (\beta_i * 16bytes) + (n-1) * \alpha_0 + \sum_{i=1}^{n-1} (\beta_0 * 16bytes)$$

onde n é o número de processadores e o processador 0 é o mestre. Os dois primeiros termos se referem às mensagens enviadas dos escravos para o mestre, que são construídas em paralelo. Os termos restantes referem-se às mensagens do mestre para os escravos. O tamanho da mensagem de controle é fixo e igual a 16 bytes no sistema Dome. Existe sobreposição de tempo nos dois últimos termos mas o valor referente a α domina este custo, dado que as mensagens de controle são pequenas.

A fase de controle do balanceamento de carga local tem custo igual a

$$FC_L = \max_{i=0}^{n-1} \alpha_i + \sum_{i=0}^{n-1} (\beta_i * 8bytes) + \max_{i=0}^{n-1} \alpha_i + \sum_{i=0}^{n-1} (\beta_i * 12bytes)$$

Os dois primeiros termos correspondem às mensagens para a esquerda e os demais às mensagens para a direita.

A diferença essencial entre as duas equações está no terceiro termo. No balanceamento global, o mestre deve enviar mensagens para todos os demais processadores. Portanto, este termo cresce linearmente com o número de processadores, o que não ocorre no balanceamento local. Isso explica uma maior escalabilidade obtida na estratégia local em relação à global (os resultados são apresentados na próxima seção).

O custo de movimentação de dados é dado por

$$FD = \max_{i=0}^{n-1} \alpha_i + \sum_{i=0}^{n-1} (\beta_i * SendToLeft[i] * tamanho) + \max_{i=0}^{n-1} \alpha_i + \sum_{i=0}^{n-1} (\beta_i * SendToRight[i] * tamanho)$$

onde $SendToLeft[i]$ e $SendToRight[i]$ são, respectivamente, o número de elementos que o processador i vai enviar para a esquerda e para a direita e *tamanho* é o número de bytes de cada elemento. Neste caso, todos os processadores constroem simultaneamente suas mensagens para a esquerda e depois para a direita. Estas equações assumem rede compartilhada (Ethernet). Em redes ATM, cujas mensagens podem ser transmitidas paralelamente, o cálculo acima é alterado mas os resultados relativos entre balanceamentos locais e globais não se alteram.

4 Resultados

4.1 Descrição dos Experimentos

Os ambientes de execução foram divididos em três grupos: estáveis homogêneos balanceados, estáveis heterogêneos e instáveis. Estabilidade refere-se à não variação da carga externa nas máquinas durante a execução do programa paralelo. Homogeneidade caracteriza um ambiente no qual todas as máquinas têm processador com a mesma velocidade. Por ambiente balanceado entende-se que as cargas são iguais em todas as máquinas, independente do processador. Os experimentos foram feitos utilizando-se um simulador do ambiente Dome implementado especialmente para este fim.

4.2 Ambiente Homogêneo Estável Balanceado

Neste ambiente as máquinas são idênticas, têm a mesma carga e são dedicadas ao experimento. Assim, a distribuição equitativa dos elementos sem o balanceamento de carga tem o melhor tempo de execução e a medida de interesse é o *overhead* de balanceamento de carga, isto é, a percentagem do tempo total de execução que é gasto exclusivamente com os processos do balanceamento de carga. A figura 2 mostra o *overhead* do balanceamento global, que é o pior caso. Como não há movimentação de dados, este custo adicional é devido ao custo de controle, que é proporcional ao número de fases e ao número de processadores (n). O *overhead* do balanceamento local é desprezível para um número pequeno de processadores, crescendo para apenas

0.06% com 64 processadores. Isto ocorre porque no balanceamento local o custo de controle é fixo por fase, isto é, não depende do número de processadores. Como não há movimentação, o custo total de balanceamento de carga é fixo para qualquer número de processadores (demais parâmetros fixos). Dado que o custo total de execução é dividido por dois cada vez que o número de processadores dobra, o custo de balanceamento local passa a ter um peso relativo maior para ambientes com grande número de processadores.

n	5 fases (%)	50 fases (%)
4	0.03	0.08
8	0.08	0.33
16	0.33	1.3
32	0.5	6.4
64	3.9	23.9
96	10.0	55.6

Figura 2: *Overhead* do balanceamento global: para 32 processadores e 50 fases, 6.4% do tempo de execução é gasto com balanceamento de carga

4.3 Ambiente Estável Heterogêneo

Este experimento inclui ambientes com processadores diferentes submetidos a cargas diversas, que se mantêm constantes durante o experimento e ambientes com processadores dedicados a um programa Dome mas com diferentes velocidades. O fator determinante é que a carga externa de cada processador não muda durante a execução do programa. Os resultados de simulação mostraram que o melhor valor para lbinit é 1 (uma) operação Dome. Isto indica que, em ambientes reais, poucas operações serão necessárias para que a heterogeneidade do ambiente seja percebida.

A figura 3 apresenta os dados de escalabilidade neste ambiente. Cada valor é o *speed-up* obtido na simulação com as condições descritas pelos parâmetros, em relação ao mesmo ambiente sem balanceamento de carga. Além de balanceamento local e global, uma outra opção foi avaliada: global na primeira fase e local nas demais (denominado balanceamento misto). Na primeira coluna, n é o número de processadores. As colunas seguintes mostram que sob uma mesma carga, o balanceamento global é ligeiramente melhor que o local para até 16 processadores. Acima de 32 processadores, o custo de balanceamento global aumenta muito. Isto ocorre porque, neste ambiente, a redistribuição ideal é obtida em uma fase no balanceamento global, uma vez que não há alteração nas cargas externas. O custo de movimentação é todo incluído na primeira fase. As fases posteriores são apenas para controle, porém este custo aumenta linearmente com o aumento do número de processadores. Neste quadro, o balanceamento misto se mostra ideal porque apresenta ganhos próximos ou superiores ao global, quando este é melhor, e também ganhos próximos ou superiores ao local, quando este é melhor. Esta regra muda para 96 processadores onde o *speed-up* do balanceamento misto é inferior ao local. Isto se deve aos custos de comunicação da primeira fase e mostra que o balanceamento local vai para a posição de melhor opção para um número grande de processadores.

No caso do balanceamento local, a redistribuição ideal é obtida gradualmente. À medida que os desbalanceamentos locais (entre vizinhos) são detectados, os dados

são movidos em cada fase. A convergência pode ou não ser obtida antes do término da execução do programa. Neste caso, o custo de movimentação é distribuído entre as fases. O que torna este ambiente escalável é o fato de que o custo de controle aumenta linearmente com o número de processadores apenas nos termos relativos a β (equação para FC_L). Observe que o *speed-up* obtido no balanceamento local diminui até 64 processadores e depois aumenta para 96 processadores. O que ocorreu neste experimento é que não houve convergência para 16, 32 e 64 processadores e houve convergência para 96 processadores na sétima fase.

Ainda na figura 3 comparando-se as colunas para as duas cargas, pode-se verificar que quanto maior a heterogeneidade no ambiente estável, mais se ganha utilizando balanceamento de carga.

parâmetros	carga: 1,2,3,1,2,3,...			carga: 1,5,9,1,5,9,...		
n	local	global	misto	local	global	misto
4	2.11	2.12	2.12	5.05	5.19	5.19
8	1.91	1.92	1.93	4.11	4.27	4.28
16	1.85	1.87	1.89	3.91	4.17	4.22
32	1.80	1.72	1.83	3.81	3.78	3.96
64	1.77	1.42	1.79	3.74	3.27	3.87
96	1.99	1.22	1.95	3.72	2.30	3.68

Figura 3: *Speed-up* no ambiente heterogêneo estável

A análise do valor ótimo para o número de operações entre fases de balanceamento, isto é, o lbmod que leva a um custo total mínimo, mostrou que este valor é dependente da heterogeneidade. Como definição inicial podemos quantificar a heterogeneidade, neste experimento, como sendo o desvio padrão dos valores (velocidade * carga) para o conjunto de processadores. Por exemplo, o desvio padrão para doze processadores com a mesma carga é 0, para doze processadores com carga {1,2,3,1,2,3,1,2,3,1,2,3} é 2.83 e para doze processadores com a carga {1,2,3,4,5,6,7,8,9,10,11,12} é 11.95.

Os resultados da análise do lbmod ótimo para balanceamento local são apresentados na figura 4. O gráfico da esquerda mostra uma curva para cada valor de heterogeneidade. O eixo y é o custo da execução em um determinado ambiente com o valor de lbmod dado pelo eixo x dividido pelo menor custo de execução para vários valores de lbmod no mesmo ambiente. O gráfico mostra que o lbmod de custo mínimo é função da heterogeneidade. Quanto maior a heterogeneidade, menor deve ser o lbmod. Para a maioria dos casos, o valor ótimo para lbmod está entre 50 e 100. Para o caso de maior heterogeneidade, o valor ótimo é 10 e para o caso mais homogêneo o valor ótimo é 500. O gráfico da direita mostra o comportamento assintótico do custo total de execução para as mesmas curvas do gráfico vizinho. Os gráficos mostram que uma escolha inadequada do lbmod pode levar a aumentos absurdos no custo total.

Foi observado também que para o mesmo conjunto de cargas, distribuições diferentes desta carga na rede de nodos altera o custo total. Por exemplo, se todos os vizinhos têm cargas diferentes, eles começam a trocar elementos desde a primeira fase de balanceamento local, o que não ocorre quando 2 processadores vizinhos têm cargas iguais. A diferença observada de custo é função do lbmod. Os resultados numéricos são apresentados em [7].

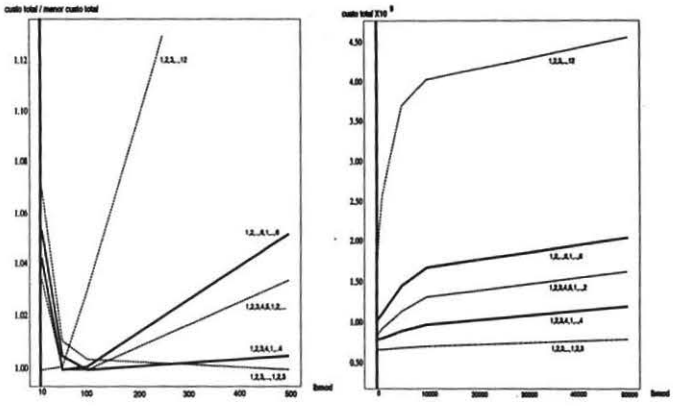

Figura 4: Avaliação do lbmod ótimo

Outra pergunta importante a ser respondida é se vale a pena transferir poucos elementos de uma máquina mais lenta para uma máquina mais rápida. A intuição diz que o custo da transferência pode ser muito grande, desaconselhando esta movimentação. Para responder a essa pergunta foi medido o custo da execução de um programa com transferência de todos os elementos recomendados pelo algoritmo de balanceamento de carga. Foram também medidos os custos de execução considerando que não haveria movimentação de dados se a quantidade de dados a ser movimentada fosse inferior a um certo limite. Este limite variou entre 0.1% e 10% do total de elementos (dimensão global). A partir dos resultados, reportados em [7], podemos concluir que os dados devem ser transferidos sempre que o algoritmo de balanceamento assim recomendar. O custo aumenta muito para todos os casos à medida que cresce a percentagem de dados não movimentados. A transferência é vantajosa quando a soma do custo para operar sobre o elemento na máquina mais rápida mais o custo de transferência é menor do que o custo de operar sobre o elemento na máquina mais lenta. É importante ressaltar que serão executadas lbmod operações sobre o elemento (e não apenas uma operação); isto implica que quanto maior for a heterogeneidade, mais se ganha fazendo a transferência . Observe que esta conclusão é função do lbmod. Quanto maior for o número de operações a ser executado sobre um elemento, melhor será sua transferência para uma máquina mais rápida.

Na avaliação da topologia da rede os resultados mostraram que a topologia em anel ganha em todos os experimentos embora este ganho seja pequeno. Num experimento com 96 processadores, a topologia anel é mais rápida cerca de 9% do que a linear.

4.4 Ambiente Instável

Neste ambiente há variação da carga nas máquinas durante a execução do programa Dome. Este é o ambiente que mais se aproxima de uma situação real. Na simulação a variação de carga foi expressa por uma variável randômica cujo valor indicou o

número médio de operações Dome executadas antes da próxima variação de carga. A figura 5 mostra o *speed-up* obtido com o balanceamento global em relação ao não balanceamento para diversas taxas de mudança de carga. λ representa o número esperado de operações Dome entre mudanças de carga. Cada resultado é a média de 50 execuções. Os valores mostram que em ambientes muito instáveis, com média de 1000 operações entre mudanças de carga e lbmod fixo igual a 50, o balanceamento obtém ganhos para até 32 processadores.

n	$\lambda = 1000$	$\lambda = 2000$	$\lambda = 10000$
4	1.39	1.42	1.49
8	1.31	1.40	1.56
16	1.27	1.45	1.74
32	1.19	1.44	1.87
64	0.78	1.04	1.61

Figura 5: Speed-up no ambiente instável para vários valores de λ

Os resultados apresentados até agora sugerem que em um ambiente de trabalho real, portanto instável, o lbmod deve ser variado dinamicamente por algum algoritmo adaptativo. Este algoritmo deve considerar dois critérios para alteração do lbmod: a heterogeneidade e a estabilidade. Inicialmente, o lbmod deve ser calculado em função da heterogeneidade. Nas fases subsequentes de balanceamento, a estabilidade deve ser avaliada; se o sistema for considerado estável então o lbmod deve ser aumentado. Caso contrário, o lbmod deve ser recalculado a partir do novo valor da heterogeneidade. No balanceamento global, o lbmod será calculado pelo mestre e enviado junto com a mensagem de retorno do mestre aos escravos. Nenhuma mensagem adicional é necessária. No balanceamento local a verificação da estabilidade, da heterogeneidade e a propagação do novo valor de lbmod para cada processador não parece ser uma questão trivial.

5 Conclusões e Trabalhos Futuros

O modelo de auto-balanceamento de programas paralelos implementado pelo Dome é interessante e relativamente simples. Os resultados obtidos por simulação são consistentes. Várias das perguntas iniciais foram respondidas em termos qualitativos e quantitativos. Uma conclusão importante é que o lbmod deve ser calculado por um algoritmo adaptativo sensível à estabilidade e à heterogeneidade do ambiente e que esta escolha deve ser cuidadosa porque o custo cresce muito para valores inadequados. Os valores ótimos para lbmod são pequenos em ambientes instáveis e maiores em ambientes estáveis. A influência da heterogeneidade foi estudada detalhadamente. Foram constatados ganhos reais com o balanceamento de carga nos diversos ambientes.

Os trabalhos futuros terão duas direções. Os resultados obtidos serão utilizados para guiar as próximas modificações no sistema implantado e validar estes mesmos resultados. Outra direção é continuar a simulação para o ambiente instável. Critérios de estabilidade deverão ser testados para viabilizar a construção do algoritmo adaptativo. Vários outros aspectos deverão ser também objeto de investigação: heterogeneidade da rede, avaliação mais cuidadosa da topologia anel e linear, opções

intermediárias entre balanceamento global e local e critérios para verificação da instabilidade.

Referências

1. Árabe, J.N.C., Beguelin, A., Lowekamp, B., Seligman, E., Starkey, M. and Stephan, P., "Dome: Parallel Programming in a Distributed Computing Environment", *Proceedings of the 10th International Parallel Processing Symposium*, Honolulu, Hawaii, April 15-19, 1996, pp. 218-224.

2. Casavant, T.L. and Kuhl, J.G., "Effects of Response and Stability on Scheduling in Distributed Computing Systems", *IEEE Transactions on Software Engineering*, Vol. SE–14, No. 11, November 1988, pp. 1578–1588.

3. Eager, D.L., Lazowska, E.D. and Zahorjan, J., "Adaptive Load Sharing in Homogeneous Distributed Systems", *IEEE Transactions on Software Engineering*, Vol. SE–12, No. 5, May 1986, pp. 662–675.

4. Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R. and Sunderam, V., *PVM: Parallel Virtual Machine — A Users' Guide and Tutorial for Networked Parallel Computing*, MIT Press, 1994.

5. Hennessy, J. L. and Patterson, D. A., *Computer Architecture: A Quantitative Approach*, Second Edition, Morgan Kaufmann Publishers,Inc., 1996.

6. Livny, M. and Melman, M., "Load Balancing in Homogeneous Broadcast Distributed Systems", *Proceedings of the ACM Comput. Network Performance Symposium.*, 1982, pp. 47-55.

7. Murta, Cristina D. and Árabe, J. Nagib C., "Estudo do Balanceamento de Carga no Ambiente Dome", Relatório Técnico 013/96, Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Brasil.

8. Shivaratri, N.G., Krueger, P. and Singhal, M., "Load Distributing for Locally Distributed Systems", *IEEE Computer*, Vol. 25, No. 12, December 1992, pp. 33–44.