Um modelo distribuído para a alocação e gerência de processadores em multicomputadores

César A. F. De Rose *
Instituto de Sistemas Operacionais
Universidade de Karlsruhe, Alemanha
Am Fasanengarten 5, 76128 Karlsruhe
Tel.: (0721) 6082645, Fax: (0721) 697760
derose@ira.uka.de

Philippe O. A. Navaux †
Instituto de Informática e CPGCC
Universidade Federal do Rio Grande do Sul
Caixa Postal 15064, 91501-970 Porto Alegre
Tel.: (051) 3368399, Fax: (051) 3365576
navaux@inf.ufrgs.br

Resumo

Diversos algoritmos para a alocação e gerência dos processadores de uma máquina paralela podem ser encontrados hoje na literatura. Grande parte destes algoritmos já fornece resultados muito bons em relação ao problema de compartilhamento de recursos, porém ainda deixam a desejar em relação a sua escalabilidade. Com o aumento do número de processadores da máquina, a estrutura centralizada utilizada atualmente por estes algoritmos torna-se rapidamente o gargalo do sistema, não permitindo a sua utilização em máquinas maciçamente paralelas. Este trabalho propõe a utilização de um modelo distribuído para a resolução deste problema e analisa o seu comportamento, tanto em nível de desempenho, quanto em nível de qualidade de resultados.

Palavras-chave: Alocação e Gerência de Processadores, Algoritmos Distribuídos, Processamento Paralelo

Abstract

Several algorithms for processor allocation in multicomputer systems are already available. Most of them have very good results in relation to the resource management problem, but lack in scalability. With the increase in the number of processors in these systems, the centralized data structure used to control the resources become very quickly the bottleneck of the system, not allowing the utilization of these algorithms in high parallel machines. This work proposes the utilization of distributed model to solve this kind of problem and analyze his performance and quality of results.

Keywords: Processor Allocation and Management, Distributed Algorithms, Parallel Processing

*Doutorando na Universidade Fridericiana de Karlsruhe; Mestrado em Ciência da Computação (CPGCC/UFRGS, 1993); Áreas de Interesse: Arquitetura de Computadores, Sistemas Operacionais, Processamento Paralelo e Distribuído; Bolsista do CNPq - Brasília

†Professor UFRGS/CPGCC; Dr. Eng. em Informática (Instituto Nacional Politécnico de Grenoble, França,1979); Áreas de Interesse: Arquitetura de Computadores, Processamento Paralelo, Avaliação de Desempenho

1 Introdução

O problema da gerência e alocação de processadores já vem sendo pesquisado faz mais de 10 anos, e diversas propostas de solução para este problema podem ser encontradas na literatura [HEI94b] [MAC94][ZHU96][DER93][MEL95].

Apesar de toda evolução ocorrida nesta área, é interessante observar que não ocorre a aplicação plena dos conhecimentos adquiridos com estes estudos nas máquinas paralelas atuais. As máquinas encontradas no mercado quase não oferecem suporte para a gerência do recurso processador, nem na forma de ferramentas, nem de um servidor em nível de Sistema Operacional.

Um dos fatores que contribui para este mau aproveitamento dos resultados das pesquisas na área são as deficiências do enfoque centralizado adotado até agora, tanto em nível de estruturação dos dados necessários para a gerência, como dos mecanismos para o seu processamento.

Este trabalho propõe uma nova abordagem para o problema, onde tanto a estrutura de dados como o controle do procedimento de gerência são distribuídos. A partir de uma análise das principais deficiências do modelo centralizado, é apresentado um modelo de gerência baseado nos conceitos de algoritmos distribuídos. Uma avaliação deste novo enfoque distribuído é realizada e vantagens e desvantagens de sua utilização são ressaltadas.

Todas as considerações e algoritmos apresentados neste trabalho assumem como máquina alvo multicomputadores, ou seja, máquinas multiprocessadoras fracamente acopladas, onde cada processador possui sua memória local e a comunicação é efetuada através de troca de mensagens. Como topologia de interconexão dos processadores será utilizada a malha (*mesh*), não só porque é a topologia mais encontrada no mercado nesta categoria de máquinas paralelas (Intel Paragon, Parsytec GC, Cray T3D), mas também por sua capacidade de se expandir em todas as dimensões. Esta característica permite uma grande flexibilidade no particionamento em sub-estruturas (partições da malha de menor dimensão), que se reflete nas estratégias de gerenciamento.

Na seção 2 é descrito o problema da gerência de processadores e na seção 3 são apresentadas as principais técnicas de gerência encontradas hoje na literatura e suas principais deficiências. A seção 4 descreve o modelo distribuído proposto, juntamente com as novas operações de gerência, e exemplos de estratégias de gerenciamento de processadores para este novo enfoque. Uma avaliação do enfoque distribuído é feita na seção 5. As conclusões deste trabalho são apresentadas na seção 6.

2 A alocação e gerência de processadores

O problema de alocação de processadores pode ser abordado em dois níveis: de tarefas e de processadores. A nível de tarefas, a alocação consiste no mapeamento de um conjunto de tarefas $\mathcal{T}$ em um conjunto de processadores $\mathcal{P}$, na forma:

$$\pi : \mathcal{T} \rightarrow \mathcal{P}$$

Para se obter uma alta taxa de utilização da máquina paralela é interessante que se permita multiprogramação, ou seja, que vários programas paralelos compartilhem a máquina. Sendo assim, a nível de processadores, a alocação consiste no mapeamento de um conjunto de programas paralelos $\mathcal{A}$ em um conjunto de todos os subconjuntos de processadores, $\wp(\mathcal{P})$:

$$\varphi : \mathcal{A} \rightarrow \wp(\mathcal{P})$$

O subconjunto $\varphi(A)$ de processadores que acomoda o programa A é chamado *território* de A. Este artigo se concentra na alocação a nível de programas, não importando quais tarefas serão mais tarde mapeadas a quais processadores. O interesse principal é a obtenção de um particionamento apropriado da máquina paralela, o que é visto como um problema de gerência de recursos. Também

é assumido que não existem duas tarefas mapeadas para o mesmo processador ao mesmo tempo, o que implica que os territórios resultantes sejam disjuntos:

$$\forall \mathcal{A}_i, \mathcal{A}_j \in \mathcal{A} : i \neq j \Rightarrow \varphi(\mathcal{A}_i) \cap \varphi(\mathcal{A}_j) = \emptyset$$

Pelo fato da interação entre tarefas ocorrer entre diferentes nodos, é interessante manter pequena a distância entre tarefas que se comunicam para diminuir os custos de comunicação. Isto é mais facilmente obtido com a alocação de territórios contíguos. Estas áreas contíguas são denominadas partições. Quando são alocadas apenas áreas contíguas, territórios e partições são idênticos.

Como a operação de alocação tem que ser eficiente para não comprometer de forma significativa o tempo de execução da aplicação, é comum que se restrinja os formatos possíveis das partições, para se obter uma certa regularidade que facilita a sua gerência. Uma estratégia de gerência é denominada *structure preserving* (preserva estrutura) se gera partições que possuem a mesma topologia da máquina paralela compartilhada. Neste caso, por exemplo, máquinas hipercúbicas são particionadas em hipercubos de menor dimensão, e malhas retangulares em sub-retângulos. A figura 1 apresenta o particionamento de uma malha de processadores para o atendimento de três requisições. As partições resultantes são disjuntas, contíguas, e mantêm a topologia da máquina compartilhada.

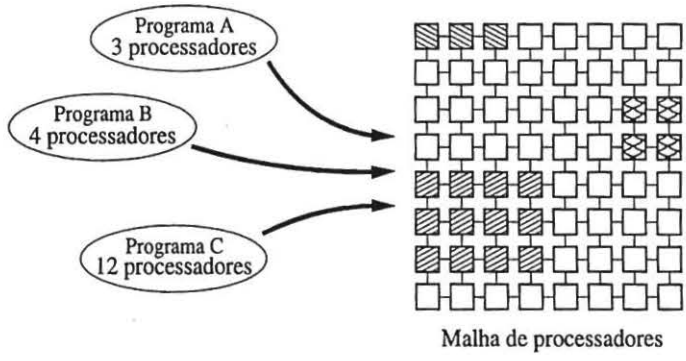

Figura 1: A operação de alocação de processadores

Embora o objetivo de qualquer estratégia de gerência de recursos seja maximizar a taxa de utilização, uma taxa de utilização de 100% do recurso processador nao é possível devido a diferentes tipos de fragmentação:

- Fragmentação externa

  Fragmentação externa inevitável
  Independente da estratégia de gerência utilizada, sempre existirá um pequeno número de processadores livres que não poderão ser alocados, porque todas as requisições que esperam por atendimento precisam mais processadores do que se encontram livres no momento, e uma alocação parcial não é efetuada.

  Fragmentação externa evitável
  Se a estratégia de gerência utilizada só alocar partições contíguas podem existir processadores livres suficientes para atender uma determinada requisição, mas, por estarem espalhados ao longo da malha, eles não podem ser alocados de forma contígua (não formam uma partição).

- Fragmentação interna

Se a estratégia de gerência restringir os possíveis formatos das partições para facilitar a gerência, talvez seja forçada a alocar uma partição maior do que foi requisitado. Neste caso, alguns dos processadores alocados ficarão sem ser usados.

Ambos os tipos de fragmentação representam desperdício de recursos, o que compromete a taxa de utilização, e portanto devem ser evitados.

Uma estratégia de gerência de processadores tem por objetivo atingir várias metas contraditórias. Primeiramente, deve evitar fragmentação para maximizar a taxa de utilização dos recursos. A seguir, mesmo não tendo informações sobre o padrão de comunicação das aplicações paralelas que ocuparão as partições alocadas, tem que se assumir que os processos destas aplicações irão se comunicar. Para minimizar os custos desta comunicação e facilitar o aproveitamento do restante da máquina é interessante que o diâmetro das partições alocadas seja mantido baixo. Outro fator que não pode ser esquecido é que pelo fato das requisições serem processadas em tempo de execução, os algoritmos de gerenciamento de recursos tem que ser rápidos.

Sendo assim, cada estratégia de gerenciamento de processadores tem que encontrar um compromisso entre as três metas abaixo, que são contraditórias entre si:

- minimizar a fragmentação (maximizar a taxa de utilização);
- minimizar o diâmetro das partições;
- minimizar o custo da gerência.

Feitas estas considerações, o problema de alocação e gerência de processadores em malhas se torna uma generalização direta do problema de alocação em memórias [HEI94a] com a diferença que enquanto a memória é um recurso unidimensional, a malha de processadores é um recurso bidimensional.

3 A alocação de processadores hoje

Os multicomputadores encontrados no mercado oferecem normalmente sistemas com suporte apenas para o particionamento manual-dinâmico dos processadores (o termo dinâmico é empregado pelos fabricantes para indicar que o particionamento pode ser alterado através de ferramentas (software)). O gerente do sistema define a divisão dos processadores da máquina em partições fixas, dependendo da carga esperada para o dia, quando a máquina é ligada e podem ser alteradas apenas por uma nova intervenção do gerente do sistema. Para fins de gerência, este tipo de particionamento é considerado estático, por não ser possível uma adaptação automática do particionamento da máquina à carga a qual está sendo submetida. Um particionamento dinâmico, por sua vez, poderia ser feito a nível de Sistema Operacional, liberando o gerente do sistema desta tarefa e melhorando consideravelmente a taxa de utilização dos processadores compartilhados.

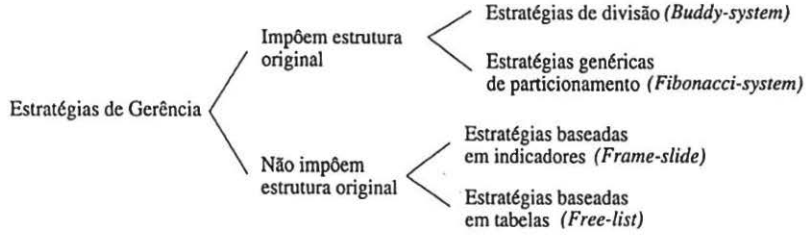

Figura 2: Estratégias de gerência encontradas na literatura aplicáveis ao caso das malhas

Na literatura podem ser encontradas diversas estratégias que procuram solucionar o problema da gerência de processadores. A figura 2 mostra um apanhado geral das técnicas que podem ser aplicadas em topologias bidimensionais, como é o caso das malhas.

Apesar de empregarem diferentes métodos para a gerência dos processadores, todas estas estratégias têm uma característica em comum, que compromete tanto o seu desempenho quanto a qualidade de seus resultados a nível de compartilhamento de recursos: o cadastro dos recursos alocados é feito com uma estrutura de dados global, que fica normalmente localizada no hospedeiro da máquina paralela. Desta forma, todas as operações de alocação e desalocação tem que ser direcionadas a este hospedeiro, centralizando a gerência dos recursos e aumentando de forma significativa o tráfego de mensagens entre hospedeiro e máquina paralela (figura 3).

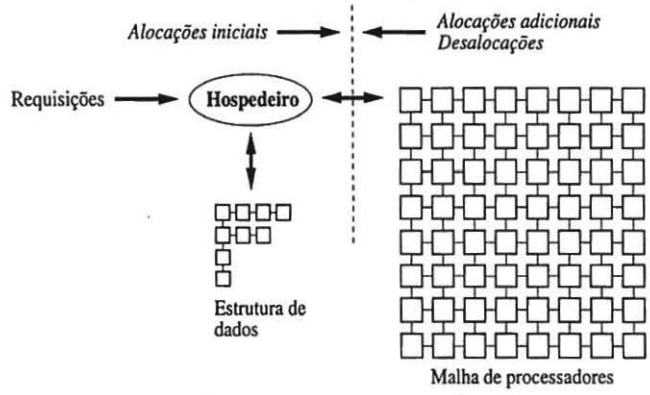

Figura 3: A gerência centralizada de processadores

3.1 As principais deficiências do enfoque centralizado

Os principais problemas encontrados nas estratégias apresentadas na figura 2 são a sua falta de escalabilidade e a geração de fragmentação.

A falta de escalabilidade resulta da utilização de estruturas centralizadas para o controle da ocupação dos processadores. Com o aumento do número de processadores a serem gerenciados, esta estrutura cresce juntamente com seu tempo de processamento, não fornecendo mais um tempo de resposta aceitável para um procedimento executado em tempo de execução. Além disto, como todas as operações de gerenciamento passam pelo hospedeiro, com o aumento do número de processadores, é intensificado o tráfego de mensagens entre hospedeiro e máquina paralela, comprometendo ainda mais esta gerência centralizada, e agravando sua condição de gargalo do sistema.

O problema da fragmentação é agravado pelo fato das estratégias simplificarem a sua gerência utilizando um particionamento que se oriente na topologia da máquina alvo (em malhas só são alocados grupos de processadores de formato retangular). Isto evita um alto tempo de resposta, mas pode causar tanto fragmentação externa (E) quanto interna (I)(ou ambas). Em [HEI94a] são medidos, através de simulação, os percentuais de fragmentação interna, fragmentação externa e taxa de utlização obtidos por algumas destas estratégias (tabela 1).

Outro fator importante é que a utilização de estruturas centralizadas na gerência dos processadores do sistema não permite um comportamento dinâmico das aplicações paralelas alocadas, no que diz respeito a variação do número de processadores utilizados pelas mesmas durante a sua execução. As aplicações poderiam desta forma reagir de forma mais flexível à sua variação de carga durante a execução, alocando e liberando processadores de forma dinâmica. Esta variação porém,

Tabela 1: Influência da fragmentação na taxa de utlização dos recursos

Estratégias de Gerência	Fragmentação(I)	Fragmentação(E)	Taxa de Util.
Tabelas disjuntas	0%	24%	76%
Divisão independente	37%	7%	56%
Divisão por área	52%	7%	41%
Divisão composta	0%	38%	62%
Divisão de Fibonacci	22%	33%	45%

acarretaria na necessidade de atualização da estrutura de dados com muito mais freqüência, aumentando de forma significativa o custo de gerenciamento. As estratégias de gerência vistas acima não permitem a variação deste número, e os processadores são alocados e liberados em bloco. Esta limitação afeta de forma significativa o compartilhamento dos processadores, resultando em uma menor taxa de utilização dos recursos.

Os motivos vistos acima são as principais causas que retardam a aplicação destas estratégias na gerência de processadores das máquinas encontradas hoje no mercado.

4 Proposta de gerência distribuída de processadores

Nesta seção será apresentada uma proposta distribuída para a solução do problema da gerência de processadores. Após uma descrição desta nova modelagem para o problema e das operações básicas envolvidas, será apresentado um algoritmo que se baseia neste modelo.

4.1 O modelo distribuído

A figura 4 apresenta uma visão global do modelo e as instâncias envolvidas na operação de gerência. As principais diferenças para o modelo centralizado (figura 3) são: (i) a inexistência de uma estrutura de dados central que contenha as informações do estado de todos os processadores da rede, e (ii) a execução das operações de gerência diretamente na malha de processadores de forma distribuída, e não em uma estrutura de dados.

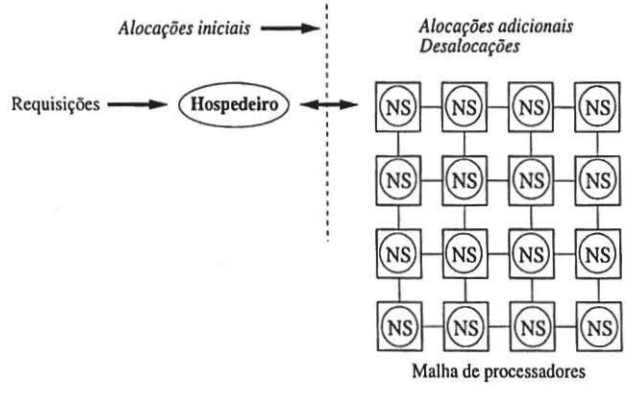

Figura 4: O modelo de gerência distribuído

Neste modelo, o hospedeiro é responsável apenas pelo recebimento de requisições e pelo disparo da procura na rede, podendo ser visto como um ponto de entrada das requisições. Como todo o processamento ocorre de forma distribuída, não existe limitação no número de pontos de entrada

para a malha de processadores, tendo que ser apenas garantido que os resultados das requisições retornem para o ponto de entrada onde foram originadas.

Outra alteração ocorre na origem das requisições. Além das requisições em bloco, no início da execução de uma aplicação, já existentes no modelo centralizado, podem agora ocorrer requisições adicionais durante a execução de uma aplicação, visando uma adaptação do número de processadores a alterações na carga de processamento das aplicações. Isto só se tornou possível porque alterações mais freqüentes no estado dos processadores, devido a estas alocações extras, não precisam ser repassadas para uma estrutura de dados central. A seção 4.2 aborda com mais detalhes os diferentes tipos de requisições e suas principais características.

Neste novo modelo, cada nodo da malha de processadores possui um núcleo do Sistema Operacional (NS) que é responsável pelas operações da gerência de processadores e sua interação. A gerência é feita desta forma através de estruturas que cooperam entre si, de forma não centralizada. Isto resulta nas seguintes alterações:

- uma visão global e atualizada da ocupação dos processadores não existe mais. Cada processador conhece a princípio o seu próprio estado. O estado de outros processadores precisa ser consultado;
- requisições ao gerente de processadores podem originar-se em qualquer um dos nodos da malha sem a necessidade do envolvimento da máquina hospedeira ;
- como a procura por processadores ocorre agora na própria malha, não é mais necessário limitá-la a partições com formas definidas (*structure preserving*);
- não existindo mais uma estrutura de dados que tenha que ser atualizada a cada operação de gerência, a operação de desalocação de um processador se resume a uma mensagem ao núcleo local de sistema do nodo em questão, marcando-o como livre.

4.2 As operações básicas do gerenciamento

Por operações básicas entende-se as operações que tratam diretamente da alocação e desalocação do recurso processador. Outras operações de gerência, como o questionamento sobre o estado de um determinado processador, ou de toda a rede, ou sobre um histórico de utilização dos recursos, não serão abordadas aqui por não se tratarem de operações críticas para o funcionamento do modelo.

Na gerência de processadores ocorrem dois tipos básicos de requisições:

1. A alocação dos processadores necessários para que uma aplicação paralela inicie o seu processamento;
2. A alocação dos processadores que são necessários durante a execução da aplicação paralela.

No primeiro tipo, a aplicação ainda não executa, não podendo requisitar ela mesma os recursos que necessita. Esta requisição se origina no gerenciador e é denominada *alocação global*. Já no segundo tipo, a aplicação já executa e pode ela mesma resquisitar os recursos adicionais que necessita. Esta requisição tem origem em um dos nodos que já foi previamente alocado à esta aplicação, e é denominada *alocação local*.

4.2.1 A alocação global

Uma aplicação deseja se utilizar dos processadores da máquina paralela, e o número de processadores que necessita para iniciar a sua execução já foi definido de alguma forma (por exemplo *Working Set* [HEI94b]).

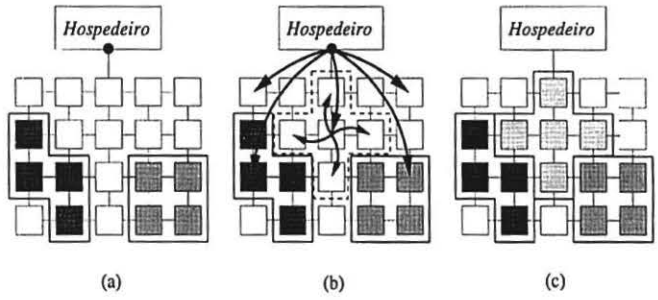

Figura 5: Alocação global de processadores: (a) requisição se origina no hospedeiro (ex. 4 processadores); (b) processadores são procurados na malha; (c) estado da malha após a alocação

A partir do módulo do gerenciador de processadores, que se encontra na máquina hospedeira, é disparada uma procura na malha de processadores pelo grupo de processadores livres desejado (figura 5). Esta procura pode ser implementada de duas formas [LYN96]:

Disparo simples: o hospedeiro realiza a procura pela malha na forma de uma onda de mensagens. Esta onda é propagada pela rede de processadores, bate nos limites da malha e retorna ao hospedeiro com o resultado da alocação.

Disparo múltiplo: vários processadores da malha (no caso extremo todos) realizam paralelamente a procura descrita acima. O hospedeiro dispara estas procuras através de uma onda de disparo e todos os processadores envolvidos executam a mesma operação. Neste caso várias ondas varrem a malha de processadores a procura de um resultado que atenda a requisição. A forma que estas ondas se influenciam umas as outras precisa ser tratada (seleção, junção, sobreposição). Depois que todas as procuras terminarem, ainda tem que ser escolhida qual das soluções encontradas, se alguma, será utilizada.

A procura com disparo simples mantém baixo o número de mensagens na rede, e simplifica o trabalho dos núcleos do sistema em cada nodo, pois eles propagam apenas uma onda. Outra vantagem é que o término do procedimento de busca é facilmente detectado (onda rebate e volta ao hospedeiro) e a solução, caso encontrada, é única. As desvantagens são o baixo grau de paralelismo obtido por uma única onda na operação de procura, e o alto tempo necessário para a varredura da malha, especialmente quando o número de processadores da máquina é grande.

Por sua vez, a procura com disparo múltiplo tem a vantagem de maior paralelização no procedimento de busca, e a conseqüente diminuição no tempo necessário para varrer toda a malha. As desvantagens são o maior fluxo de mensagens e a maior complexidade dos núcleos do sistema nos nodos, que agora lidam com diversas ondas. Além disto, é mais difícil detectar o término do procedimento e ainda existe a possibilidade de ter que se escolher uma entre diversas soluções.

4.2.2 A alocação local

No caso de uma aplicação que já esta sendo executada desejar alocar processadores extras para adaptar seu número de processadores dinamicamente à sua carga de trabalho, ela pode fazê-lo com uma alocação local.

A aplicação envia uma requisição de alocação local para um dos núcleos de sistema dos nodos que utiliza. Este núcleo dispara, então, a procura na rede para encontrar processadores livres. Como neste caso a vizinhança é necessária, o escopo da procura se limita a nodos vizinhos da área que já havia sido previamente alocada para esta aplicação (figura 6). Neste caso não se justifica a

complexidade de uma procura por disparo múltiplo, sendo utilizada uma implementação com uma única onda (disparo simples).

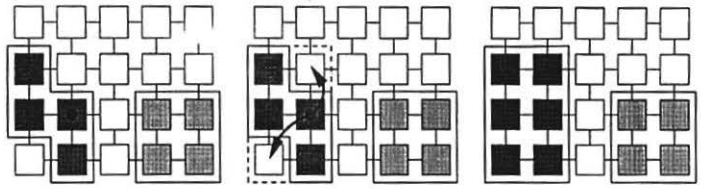

Figura 6: Alocação local de processadores: (a) requisição se origina localmente em um nodo (ex. mais 2 processadores); (b) os processadores são procurados ao redor dos já alocados anteriormente; (c) estado da malha após a alocação

Este tipo de alocação é feita de forma extremamente eficiente, gerando poucas mensagens devido ao pequeno escopo de procura. Outro fator importante é que este tipo de alocação pode se aproveitar de nodos livres isolados, que não poderiam ser usados para atender uma alocação global, diminuindo assim a fragmentação externa do gerenciamento e melhorando a taxa de utilização dos processadores da rede.

4.2.3 A operação de desalocação

A operação de desalocação no modelo centralizado implica na atualização da estrutura de dados que faz o controle dos processadores, para que os nodos fossem novamente vistos como livres na próxima operação de alocação. Dependendo do tipo de estratégia utilizada, a operação de desalocação pode vir a ser mais complexa que a própria operação de alocação, devido ao procedimento de recompactação de pequenas áreas livres em áreas maiores.

No modelo distribuído, como não existe mais a estrutura de dados central para ser atualizada, a operação de desalocação se resume ao envio de uma mensagem local do nodo em questão para o seu núcleo do sistema. O núcleo registra este nodo como livre e o nodo já se encontra disponível quando for atingido pela próxima onda de procura.

4.3 Exemplos de algoritmos distribuídos

A seguir são descritos dois algoritmos que exemplificam como pode ser feita a gerência do recurso processador de forma distribuída.

4.3.1 Algoritmo *frame-slide*

Este algoritmo é uma adaptação do algoritmo centralizado de mesmo nome [CHU91] para o modelo de gerência distribuído. O princípio básico do algoritmo é mantido: na alocação global de processadores uma janela retangular (*frame*) que contém o número de processadores desejados pela requisição é deslocada (*slide*) sobre a estrutura de dados onde se encontram os estados dos processadores até que uma área livre seja encontrada (figura 7a).

No caso distribuído, esta janela usada na procura é deslocada diretamente sobre a malha de processadores e sua implementação é feita através de 2 tipos de ondas (figura 7b) usando o princípio de disparo simples:

Onda Pivô: é a onda básica de procura que varre toda a malha até encontrar um nodo livre que será o canto superior esquerdo da janela em questão (nodo pivô). Quando encontrado,

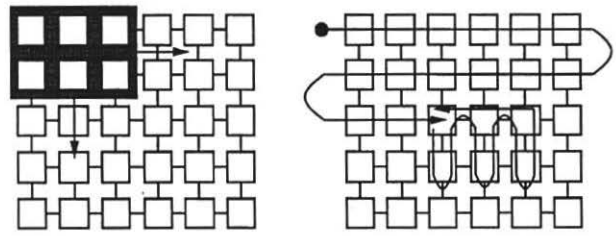

Figura 7: Algoritmo frame-slide distribuído

é disparado neste nodo uma segunda onda denominda *vertical-horizontal*. Se esta segunda onda retornar ao nodo pivô com um resultado positivo, o procedimento pode ser interrompido pois o retângulo desejado foi encontrado. Caso contrário é dada continuidade a onda pivô;

Onda Vertical-Horizontal: esta onda, disparada a partir do nodo pivô, varre a malha primeiramente procurando nodos livres no sentido vertical para depois avançar a procura no sentido horizontal. Enquanto forem sendo encontrados nodos livres eles vão sendo marcados como alocados até que o retângulo esteja completo ou que um nodo já alocado seja encontrado. No primeiro caso, a onda retorna um valor positivo, no segundo, negativo.

O algoritmo é extremamente simples, utilizando-se de uma estratégia de alocação baseada em partições regulares (retângulos), e não permitindo alocações locais. O procedimento de desalocação é trivial, sendo o nodo em questão apenas marcado como livre, o que é feito com uma mensagem ao seu núcleo do sistema local. O procedimento de procura na malha, que é implementado com ondas de disparo simples e de forma seqüencial, pode ser muito custoso, especialmente em malhas grandes com muitos processadores alocados.

4.3.2 Algoritmo de escoamento

Este algoritmo se baseia no princípio do escoamento de água. De um ponto de origem escoa uma determinada quantidade de água. A água só pode escoar para as direções onde não encontra resistência. Um fator importante é que a água que escoa tem uma certa coesão, o que faz que a poça resultante se mantenha o mais concêntrica possível.

No caso da alocação distribuída de processadores, a quantidade de processadores que se deseja alocar representa a quantidade de água que escoa, os processadores já alocados na malha representam as áreas de resistência, e a área final formada pelos processadores encontrados para atender a requisição, a poça de água resultante.

A implementação distribuída do algoritmo de escoamento é composta pelas seguintes fases:

1. o ponto de origem do escoamento é procurado na malha através de uma onda seqüencial semelhante a onda pivô do algoritmo *frame-slide* (4.3.1). Todos os nodos livres encontrados ao longo desta onda são utilizados seqüencialmente como ponto de origem. Se nenhum deles resultar em uma área que atenda a requisição, ou nenhum processador livre for encontrado, o pedido de alocação é negado;

2. todos os vizinhos de um ponto de origem (4 se não for borda) são questionados em paralelo se se encontram desalocados (direções sem resistência para o escoamento de água são procuradas);

3. a carga restante a ser distribuída (a carga original já foi decrescida do processador que representa o ponto de origem) é dividida entre os processadores vizinhos que se encontram livres. Se todos os vizinhos livres receberão carga, e como esta carga será dividida entre os escolhidos, são questões que afetam a área resultante de forma significativa (aqui são aplicadas heurísticas para se obter as características físicas desejadas, como por exemplo a coesão);

4. cada um dos vizinhos livres que recebe carga reinicia de forma recursiva na fase 2 como se fosse o ponto de origem (o processamento dos vizinhos ocorre em paralelo);

5. confirmações e custos são coletados e avaliados no ponto de origem. Se a distribuição de carga inicial não puder ser atendida, nenhuma outra é experimentada no mesmo ponto de origem. O ponto de origem em questão é liberado e é dada continuidade a procura seqüencial da fase 1;

6. a resposta é enviada para o ponto de entrada.

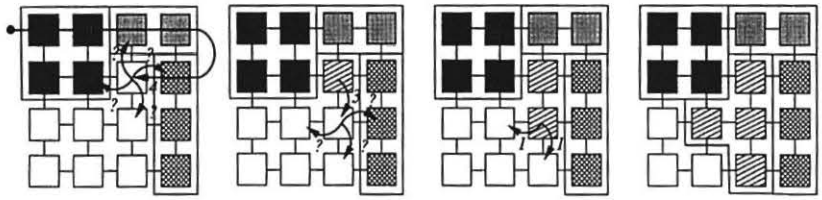

Figura 8: Fases do algoritmo de escoamento

A figura 8 exemplifica o atendimento de uma requisição de 4 processadores. Após a obtenção de um ponto de origem (figura 8a) são verificadas as possíveis direções de escoamento e a carga restante é redistribuída (figura 8b,c). A figura 8d apresenta o estado da malha após a operação de alocação.

O algoritmo se utiliza de uma estratégia de alocação que se baseia em partições de forma livre, dando maior flexibilidade a operação de procura de grupos de processadores livres na grade. Esta primeira versão do algoritmo não implementa alocações locais e a operação de desalocação é implementada de forma análoga ao algoritmo *frame-slide*.

4.4 Mecanismos utilizados para obtenção de maior desempenho

Após a avaliação dos resultados obtidos pelas primeiras versões dos algoritmos distribuídos apresentados acima, foram identificadas características do modelo distribuído de alocação que poderiam ser melhor exploradas para aumentar o desempenho desta nova classe de algoritmos.

4.4.1 Múltiplos pontos de entrada

O fato do modelo centralizado de alocação possuir apenas um ponto de comunicação entre hospedeiro e máquina paralela não resulta em maiores problemas já que o gargalo se encontra no processamento das estruturas de dados que fazem o cadastro dos recursos disponíveis, que se concentram na máquina hospedeira.

Com a quebra deste gargalo no modelo distribuído e a transferência do processamento de alocação para a própria malha de processadores, esta questão passa a ter um papel importante na distribuição de carga entre os processadores envolvidos na operação de alocação, tanto a nível da carga gerada pela operação de gerência como da carga das aplicações.

Um único ponto de entrada para as ondas de procura concentra um maior fluxo de mensagens de gerência em torno de uma única região da malha e faz com que os processadores e canais de comunicação ao redor deste ponto recebam bem mais carga que os demais. Outro efeito negativo é o aumento do tempo médio de alocação com o aumento da carga alocada. Devido a política de alocação *first-fit*, as primeiras requisições atendidas são alocadas ao redor deste ponto de entrada, gerando um aumento gradativo no tempo de procura por regiões livres, que se encontram cada vez mais distantes do ponto de entrada.

Com um aumento do número de pontos de entrada e seu posicionamento estratégico (os melhores resultados foram obtidos com um ponto de entrada em cada canto da malha) é possível distribuir melhor a carga entre os processadores da malha e diminuir a distância entre a origem das ondas de procura e as áreas livres da malha de processadores. Cada nodo da malha que abriga um ponto de entrada necessita uma conexão física com a máquina hospedeira, na forma de um canal de comunicação.

4.4.2 Paralelismo entre requisições

No modelo centralizado a exploração de paralelismo entre requisições esbarra na consulta às estruturas de dados do hospedeiro, que efetuam o cadastro dos recursos disponíveis. Como a operação de alocação efetua consultas e alterações nestas estruturas, era necessário um tratamento como zona crítica e sua proteção de acesso, com por exemplo, com a utilização de semáforos.

No modelo distribuído estas informações estão distribuídas pela rede, e o acesso ao estado de cada processador é encapsulado pelas operações de um núcleo de sistema local a este nodo. Desta forma se torna possível a consulta paralela de diferentes regiões da malha por diferentes requisições.

Através da exploração deste tipo de paralelismo é possível reduzir o tempo médio de atendimento das requisição e aumentar a taxa de utilização da máquina. Os melhores resultados são obtidos quando a exploração do paralelismo entre requisições é combinada com a utilização de múltiplos pontos de entrada (4.4.1). Desta forma operações de procura podem ser disparadas em paralelo em diferentes regiões da malha de processadores, e no melhor caso possível, ocorrer sem interferência mútua.

5 Os resultados obtidos

Com o objetivo de avaliar o modelo proposto, os algoritmos descritos na seção 4.3 foram implementados sob o sistema operacional COSY [BUT97][DER95], e resultados foram medidos em um ambiente que simula as condições reais de uso. As tabelas 2 e 3 apresentam os resultados obtidos em uma máquina *Mega Cluster* com 64 nodos Transputer configurados como malha de tamanho 16 (4x4) e 64 (8x8), respectivamente. Em ambos os casos foram tratadas 100 requisições de curta duração, com tamanho médio de 5 (máximo 16) processadores na tabela 2, e de 20 processadores (máximo 32) na tabela 3.

Tabela 2: Resultados medidos com 16 processadores (malha de 4x4)

Algoritmo	*par*	t (s)	%	Falhas	Msgs.	%	Frag. I.	Frag. E.	Taxa de Util.
sframe	-	14,27	100	39	4315	100	4,97%	51,09%	33,46%
smframe	-	9,73	68,2	38	3997	92,6	4,97%	49,81%	33,77%
amframe	1	13,12	91,9	47	3629	84,1	4,53%	50,34%	27,87%
	3	12,20	85,5	52	4235	98,2	4,53%	45,86%	30,07%
smesco	-	10,01	70,2	30	9753	226	0%	23,13%	42,04%
amescoa	1	7,28	51,0	3	3955	91,7	0%	37,32%	34,14%
	3	6,41	44,9	7	4694	108,8	0%	34,38%	36,12%

Tabela 3: Resultados medidos com 64 processadores (malha de 8x8)

Algoritmo	*par*	t (s)	%	Falhas	Msgs.	%	Frag. I.	Frag. E.	Taxa de Util.
sframe	-	19,20	100	39	28344	100	8,75%	58,09%	22,82%
smframe	-	14,60	76,0	35	18555	65,5	8,75%	56,91%	22,75%
amframe	1	13,65	71,0	38	16350	57,7	10,02%	56,56%	14,76%
	3	11,78	61,4	43	19520	68,9	10,02%	50,37%	17,92%
smesco	-	11,22	58,4	6	34799	122,8	0%	43,89%	26,91%
amescoa	1	7,88	41,0	0	15918	56,2	0%	0%	27,76%
	3	6,74	35,1	4	17623	62,2	0%	0%	31,32%

As abreviações **frame** e **escoa** indicam o algoritmo utilizado, *frame-slide* e de escoamento, respectivamente. O prefixo **m** indica uma versão otimizada com a utilização de múltiplos pontos de entrada (resultados medidos com 2 pontos). Os prefixos **a** e **s** indicam versões dos algoritmos com e sem a otimização através do aproveitamento de paralelismo entre requisições (a coluna **par** indica o número máximo de requisições em paralelo). A coluna **Falhas** indica o número de requisições que foram negadas e tiveram que ser repetidas. A coluna **Msgs** o número total de mensagens geradas pelo algoritmo.

O algoritmo *frame-slide* foi escolhido como base de comparação entre os diferentes algoritmos distribuídos e suas otimizações. A relação entre alguns de seus resultados e os resultados obtidos pelas demais versões são destacados em uma comparação percentual (colunas %).

Os resultados demonstram que a maior flexibilidade do algoritmo de escoamento em encontrar áreas de processadores livres na malha (partições de forma livre) diminui consideravelmente a fragmentação externa e elimina a interna. Isto significa uma maior eficácia no atendimento de requisições, diminuindo o tempo de execução do algoritmo, número de mensagens por ele geradas, e o número de requisições negadas (**Falhas**). Este aumento de complexidade na operação de procura por partições se reflete diretamente no número de mensagens geradas pelo algoritmo (acréscimo médio de 74%). Com a utilização de múltiplos pontos de entrada, estas mensagens são melhor distribuídas pelos nodos da malha, amenizando a sobrecarga dos canais de comunicação.

Para destacar as qualidades do algoritmo de escoamento é apresentado na figura 9 o seu comportamento em relação a carga a ser gerenciada. A variação da carga foi obtida com um aumento gradual do tamanho médio das partições requisitadas em uma malha com 64 processadores. O número de requisições a serem tratadas foi aumentado para 200 e as medições foram feitas sob carga máxima (todas as requisições tem tempo de chegada igual a zero e esperam para ser atendidas). Isto aumenta consideravelmente a carga a qual a máquina é submetida pois são evitados períodos onde não existem requisições a serem tratadas.

Na figura 9a é analisada a qualidade da gerência, com os valores da taxa de utilização, fragmentação externa (interna não existe) e o acréscimo ocorrido no diâmetro das partições, todas expressas em valores percentuais. No gráfico é possível observar que a taxa de utilização da máquina se mantém extremamente alta com uma fragmentação externa quase constante, mesmo com um aumento significativo do tamanho médio das partições gerenciadas. Isto demonstra a ótima capacidade do algoritmo de escoamento de encontrar e de se aproveitar de regiões irregulares de processadores livres. Isto é obtido com a flexibilidade da forma das partições, o que pode resultar em partições com diâmetros bem maiores do que as de forma regular, um efeito que não seria desejável (ver metas seção 2). A terceira curva (diâmetro da partição) indica o acréscimo percentual que as partições de forma livre obtiveram em seu diâmetro quando comparadas com partições de forma regular. Em relação a melhoria obtida na qualidade da gerência, o acréscimo médio de 17% no diâmetro pode ser considerado muito bom.

Na figura 9b é analisado o desempenho do algoritmo, com o número de mensagens geradas e o tempo necessário para o tratamento das 200 requisições. O número de mensagens varia de 30 a 300

mil e se refere a mensagens entre vizinhos (distância igual a 1). O tempo de execução medido varia de 22 a 75 segundos. Apesar da alta taxa de crescimento no número de mensagens geradas pelo algoritmo é interessante verificar que o aumento no tempo de execução foi pequeno. Isto resulta do alto grau de paralelismo do algoritmo e de sua execução de forma distribuída. Outro fator importante é que a carga de mensagens é distribuída de forma homogênea entre os nodos da malha, em função dos múltiplos pontos de entrada utilizados.

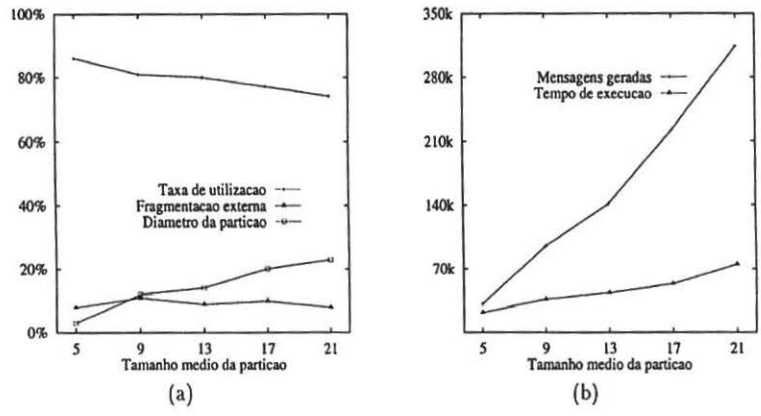

Figura 9: Comportamento do algoritmo de escoamento com a variação da carga gerenciada

Ambas as otimizações propostas em 4.4 resultaram em ganho de desempenho para os algoritmos. Os múltiplos pontos de entrada diminuíram o tempo médio de atendimento de uma requisição e melhoraram a distribuição de carga na malha, como era esperado. Isto resultou na queda no tempo total de execução do algoritmo e em um menor número de mensagens geradas. O atendimento em paralelo de múltiplas requisições por sua vez contribuiu para que o tempo médio de espera por atendimento fosse reduzido, e a carga a qual a máquina é submetida aumentada (várias requisições tem que ser gerenciadas ao mesmo tempo). Isto resultou em uma diminuição no tempo total de execução do algoritmo e um aumento da taxa de utilização da máquina.

Como a utilização de paralelismo entre requisições pode alterar a ordem de tratamento das requisições, a comparação destas versões com as demais tem que ser feita com cautela. Para facilitar este tipo de comparação foram incluídos nestes casos também resultados com par igual a um, ou seja, sem atendimento em paralelo.

Os resultados apresentados na tabela 3 demonstram a escalabilidade dos algoritmos distribuídos e o potencial do modelo aqui apresentado. Mesmo a quadruplicação do número de processadores gerenciados e do tamanho de médio de partições alocadas, não alterou de forma significativa a qualidade dos resultados obtidos.

6 Conclusões

Neste trabalho é proposta a utilização de técnicas de processamento distribuído para a melhoria do procedimento de gerência de processadores em multicomputadores.

Um modelo distribuído de gerência de processadores é proposto, onde não existe mais a figura de uma estrutura central de controle do estado dos processadores, e as operações de gerência são aplicadas diretamente na malha de processadores. Desta forma, não existe mais um cadastro global do estado da rede e os próprios processadores são responsáveis pela obtenção de soluções, através de comunicação e cooperação.

Para ilustrar a utilização do modelo distribuído são redefinidas as operações básicas da gerência de processadores para este novo ambiente, e duas estratégias de gerência distribuídas são apresentadas e avaliadas, demonstrando o potencial do modelo apresentado.

Neste novo enfoque é possível uma maior paralelização das operações de alocação, a eliminação dos gargalos do modelo centralizado, e uma melhoria na escalabilidade das estratégias de gerência de processadores. Desta forma, obtém-se uma melhora na relação da qualidade dos resultados obtidos com o tempo gasto na gerência. Isto significa que estratégias mais complexas podem ser aplicadas em um problema que tem que ser resolvido em tempo de execução, melhorando a taxa de utilização dos processadores e a qualidade de sua gerência, sem comprometer de forma significativa o tempo de resposta do sistema.

É importante ressaltar porém, que os algoritmos distribuídos, por sua vez, também sofrem limitações em sua complexidade devido a geração de tráfego de mensagens de gerenciamento. Estas mensagens disputam os mesmos canais usados para a troca de mensagens das aplicações que executam na máquina. Como um algoritmo mais complexo gera normalmente um maior fluxo de mensagens na rede, é importante observar que a rede não seja sobrecarregada, e que desta forma, o desempenho das aplicações que executam na máquina seja afetado de forma considerável por causa do gerenciamento.

É de se esperar que esta alteração significativa na forma de gerenciar processadores permita o desenvolvimento de algoritmos, que por suas qualidades, venham a ser efetivamente incorporados ao software básico dos multicomputadores com um grande número de processadores. Isto seria, sem dúvida, um passo importante para que a gerência de processadores venha ser um dia tão eficaz e transparente, como é hoje a gerência de memórias.

Referências

[BUT97] BUTENUTH, R.; HEISS, H. U. Small, Scalable, and Efficient Microkernels for Highly Prallel Computers are possible: COSY as an Example. In: International Conference on Advances in Parallel and Distributed Computing, 1997, Shangai, China. *Proceedings* Shangai, China: 1997.

[CHU91] CHUANG, P. J.; TZENG, N. F. An Efficient Submesh Allocation Strategy for Mesh Computer Systems. In: Conf. DCS, 11, 1991, Arlington, TX. *Proceedings* Arlington, TX: 1991. p. 256-263.

[DER93] DE ROSE, C.A.F.; NAVAUX, P. Algoritmos paralelos para a gerência e alocação de processadores em máquinas multiprocessadoras hipercúbicas. In: Simpósio Brasileiro de Arquitetura de Computadores - Processamento Paralelo, 5.,1993, Florianópolis. *Proceedings* Florianópolis: SBC, 1993. 775 p., p. 459-474.

[DER95] DE ROSE, C.A.F.; NAVAUX, P. Avaliação de desempenho do sistema COSY: um sistema operacional concorrente para Transputers. In: Simpósio Brasileiro de Arquitetura de Computadores - Processamento Paralelo, 7., 1995, Canela. *Proceedings* Canela: SBC, 1995. 639 p., p. 383-397.

[HEI94a] HEISS, H. U. Dynamic Partitioning of Large Scale Multicomputer Systems. In: Conference on massively Parallel Computing Systems (MPCS'94), 1994, Ischia. *Proceedings* Ischia, 1994.

[HEI94b] HEISS, H. U. *Alocação de processadores em Máquinas Paralelas.* BI- Verlag Manheim, Reihe Informatik Band 98 (1994 - Em Alemão).

[LYN96] LYNCH, N. A. *Distributed algorithms.* (The Morgan Kaufmann Series in Data Management Systems) San Francisco, Calif. : Kaufmann , 1996. 872 P.

[MAC94] MCCANN, C.; ZAHORJAN, J. Processor Allocation Policies for Message-Passing Parallel Computers. *ACM SIGMETRICS Performance Evaluation Review*, vol 22, n.1, p:19-32, 1994.

[MEL95] MELHART, B., MORGENSTERN, C., NUTE, T. A compendium of processor allocation strategies for two-dimensional mesh connected systems. *Concurrency:Practice and Experience*, vol 7, n.5, p:497-514, 1995.

[ZHU96] ZHU, Y. Fast processor allocation and dynamic scheduling for mesh multicomputers. *International Journal of Computer Systems Science and Engineering*, vol 11, n.2, p:99-107, 1996.