Balanceamento de Carga em Ambientes Distribuídos

Isabel Cristina Mello Rosseti
e-mail: rosseti@inf.puc-rio.br

PUC-RioInf.MCC27/97 Agosto, 1997

Abstract
Many load balancing techniques have been proposed, in order to obtain improvement in the performance of parallel applications. This paper presents some existing load balancing algorithms. These algorithms are presented according to an analytical model called matrix iterative model, that was chosen for the variety of load balancing algorithms which it may represent.

Keywords
Load balancing, load balancing algorithms, matrix iterative model, analytical model.

Resumo
Diversas técnicas de balanceamento de carga têm sido propostas, a fim de obter melhoria no desempenho de aplicações paralelas. O objetivo deste trabalho é apresentar uma descrição de alguns algoritmos de balanceamento de carga existentes. Estes algoritmos são apresentados segundo um modelo analítico chamado de modelo iterativo de matriz. Este modelo analítico foi escolhido pela variedade de algoritmos de balanceamento de carga que ele pode representar.

Palavras-chave
Balanceamento de carga, algoritmos de balanceamento de carga, modelo iterativo de matriz, modelo analítico.

1 Introdução

Para obter benefícios com a computação paralela, a carga computacional de uma aplicação deve ser distribuída de maneira efetiva entre todos os processadores do sistema [FG96]. Essa distribuição de carga computacional é chamada de balanceamento de carga.

O balanceamento de carga é o processo que permite melhor compartilhamento dos recursos computacionais das diversas máquinas interligadas pela redistribuição dessa carga através dos nós que compõem o sistema.

Uma das representações usadas para estudar o desempenho de algoritmos de balanceamento de carga é o modelo analítico [ELZ86]. Segundo [Chah80], o modelo analítico é uma representação do sistema onde é possível deduzir as fórmulas ou equações matemáticas que expressam uma solução para um certo problema em estudo. Como muitos dos modelos analíticos existentes são baseados em teoria das filas [MD94], eles geralmente não conseguem derivar resultados gerais aplicáveis a um grande intervalo de procedimentos de balanceamento de carga, porque utilizam aproximações simplificadas para deduzir as equações matemáticas que eles usam, o que limita o número de algoritmos de balanceamento de carga que podem fazer uso deles. Além disso, esses modelos levam a análises de desempenho irreais devido a essas aproximações [MD94]. Esses modelos são úteis, entretanto, quando se quer fazer uma análise geral do comportamento do sistema.

O Modelo Iterativo de Matriz, definido em [Cybe89], é um modelo analítico em que as deduções matemáticas estão todas baseadas em matrizes que representam o sistema. Como esse modelo independe do algoritmo de balanceamento de carga utilizado [FG96], ele pode ser usado para representar uma grande variedade de procedimentos de balanceamento.

O objetivo desse trabalho é apresentar um survey de alguns algoritmos de balanceamento de carga existentes. Esses algoritmos serão apresentados segundo o Modelo Iterativo de Matriz.

Esse trabalho é organizado como segue. Na seção dois, será discutido o balanceamento de carga em ambientes distribuídos. Na seção três, o Modelo Iterativo de Matriz será mostrado. Na seção quatro, os algoritmos de balanceamento de carga, apresentados segundo o Modelo Iterativo de Matriz serão mostrados. Na seção cinco, será apresentada as considerações finais sobre o trabalho.

2 Conceitos Básicos de Balanceamento de Carga

Um problema importante que aparece em sistemas distribuídos é o de melhorar o desempenho de um conjunto de nós através do balanceamento da carga do sistema.

O balanceamento de carga em um sistema distribuído é o processo de compartilhar os recursos computacionais das diversas máquinas interligadas pela redistribuição da carga através dos nós do sistema.

Um esquema de balanceamento de carga pode ser classificado, conforme sua relação com o estado atual de carregamento do sistema, em estático, dinâmico e adaptativo. Os esquemas de balanceamento de carga estáticos independem do estado do sistema. Os dinâmicos sempre consideram o estado do sistema e os esquemas adaptativos mudam seus parâmetros ou políticas para refletir as mudanças do estado do sistema [LLC96].

Três componentes importantes do balanceamento de carga adaptativo e dinâmico são: a política de transferência, a política de informação e a política de localização. A política de transferência determina se uma tarefa deve ser transferida. A política de informação determina quais informações serão usadas para fazer as redistribuições de carga e como elas serão distribuídas entre os participantes. Isto é, determina as regras para coletar informações sobre os nós participantes do sistema. Finalmente, a política de localização é responsável por achar os nós de destino para onde as tarefas serão transferidas [LLC96].

3 Modelo Iterativo de Matriz

O Modelo Iterativo de Matriz, proposto em [FG96], é um modelo analítico que pode ser usado para representar um grande intervalo de algoritmos de balanceamento de carga. O modelo descreve a mudança na distribuição de tarefas sobre os processadores devido ao algoritmo de balanceamento de carga.

Para desenvolver esse modelo assume-se que:

(a) a aplicação pode ser dividida em um grande número de tarefas (comparado ao número de processadores);

(b) as tarefas são idênticas, com relação aos custos computacionais e de comunicação, e podem ser executadas por qualquer processador disponível;

(c) os processadores podem ter diferentes poderes computacionais.

O conjunto de processadores disponíveis é denotado por $P = \{P_1, ..., P_p\}$, onde p é o número de processadores. O poder computacional do processador P_i é denotado por C_i, onde C_i é proporcional ao número médio de operações executadas pelo processador em uma unidade de tempo.

O tempo gasto para executar uma tarefa em P_i é inversamente proporcional a C_i. Para encontrar o balanceamento de carga em um sistema heterogêneo, aloca-se a cada processador um número de tarefas proporcional ao respectivo poder computacional. Por exemplo, se P_i é duas vezes mais rápido que P_j (isto é, $C_i = 2C_j$), aloca-se a P_i o dobro de tarefas de P_j, para o sistema ficar balanceado.

O algoritmo de balanceamento de carga é invocado em instantes de tempo denotados por k ($k = 1, 2, 3, ...$). O algoritmo deve ser invocado em intervalos regulares, no fim de cada iteração em aplicações iterativas, ou deve ser disparado por um evento externo.

O Modelo Iterativo de Matriz descreve a distribuição das tarefas de cada processador do sistema, em cada um dos instantes k, quando o algoritmo de balanceamento de carga é invocado. O estado do sistema em k é representado por um vetor de distribuição de tarefas $w(k)$, onde $w_i(k)$ é o número de tarefas no processador P_i em k.

A distribuição de tarefas no instante de tempo $(k + 1)$, $w(k + 1)$, pode ser expressa como função de:

1. $w(k)$, a distribuição de tarefas em k;

2. a transferência de tarefas devido ao balanceamento de carga;

3. a chegada de tarefas antes da *(k + 1)-ésima* invocação do algoritmo de balanceamento de carga;

4. a saída de tarefas antes da *(k + 1)-ésima* invocação do algoritmo de balanceamento de carga.

O modelo de transferência dinâmico para cada processador, P_i, pode ser representado por:
$w_i(k+1) = w_i(k) - \textit{tarefas enviadas por } P_i + \textit{tarefas recebidas por } P_i + \textit{chegada de tarefas durante } [k, k+1] - \textit{saída de tarefas durante } [k, k+1]$

{1}

O algoritmo de balanceamento de carga tenta transferir as tarefas para compensar o desequilíbrio resultante da mudança dinâmica da aplicação e pode ser descrito em termos das três políticas seguintes:

1. política de informação $\rightarrow$ os algoritmos de balanceamento de carga diferem na informação de estado do sistema que eles necessitam e também diferem na aquisição dessa informação. Alguns algoritmos exigem uma medida de carga média em todos os processadores (ou num subconjunto de nós) do sistema, enquanto outros exigem a distribuição de carga atual dos processadores do sistema.
   Define-se $\bar{w}(k)$ como o vetor que representa a distribuição de carga ideal, onde o número de tarefas em cada processador é proporcional ao seu poder computacional. Define-se, também, $\tilde{w}(k)$ como o vetor que representa uma versão aproximada da distribuição ideal.

2. política de transferência $\rightarrow$ quando um algoritmo de balanceamento de carga é invocado, cada processador decide se ele quer participar do balanceamento de carga, baseado na informação de estado disponível. Em algoritmos *Sender-Initiated*, os processadores participam no balanceamento de carga se sua carga é maior que o *threshold* de tarefas $w^*(k)$. A participação do processador no balanceamento de carga em k é representado pela matriz $A(k)$. Se o processador i decide participar do balanceamento de carga, o elemento A_{ii} recebe o valor um; todos os outros elementos recebem o valor zero.

3. política de localização $\rightarrow$ cada processador que decide participar do balanceamento de carga tem de calcular:

   (a) o número de tarefas transferidas;

   (b) os destinos das tarefas.

   As decisões de migração de tarefas são representadas por uma matriz pxp, $M(k)$, onde M_{ij} é a fração de tarefas que o processador P_i decide enviar a P_j (isto é, P_i envia $M_{ij}(k)w_i(k)$ tarefas a P_j). As decisões de migração de tarefas ficam necessariamente restritas à estrutura imposta pelo algoritmo de balanceamento de carga ou pela topologia do sistema. Define-se H como a matriz de topologia. H é uma matriz pxp que descreve os caminhos de migração possíveis. O elemento H_{ij} é igual a um, se os processadores P_i e P_j podem trocar tarefas. Caso contrário, $H_{ij} = 0$. Portanto, na construção da matriz M, $M_{ij} \neq 0$ se e somente se $H_{ij} = 1$.

A equação {1} pode ser reescrita como segue:
$w_i(k+1) = w_i(k) - \sum_{j\neq i} M_{ij}(k)A_{ii}(k)w_i(k) + \sum_{j\neq i} M_{ji}(k)A_{jj}(k)w_j(k) + \lambda_i(k) - \mu_i(k)$

{2}

onde $\lambda(k)$ e $\mu(k)$ são vetores que denotam a taxa de chegada e a taxa de saída, respectivamente.

As duas somatórias representam a atividade de balanceamento de carga. A primeira corresponde à soma de todas as tarefas transferidas de P_i para os outros processadores e a segunda representa a soma de todas as tarefas transferidas para P_i.

A distribuição "ideal", $\bar{w}(k)$, representa a distribuição de tarefas em um estado de balanceamento ideal de carga, $\bar{w}_i(k)$. A carga "ideal" no processador P_i é dada por:
$\bar{w}_i(k) = \frac{C_i}{\sum_{\forall j P_j} C_j} \sum_{\forall j P_j} w_j(k)$

$$\{3\}$$

O primeiro fator da multiplicação é a fração do total de computação associado ao processador P_i e o segundo fator, $\sum_{\forall j P_j} w_j(k)$, é o número total de tarefas no sistema.

Muitos algoritmos usam a informação local (isto é, a distribuição de tarefas nos processadores da vizinhança), como uma aproximação da informação de estado global. A vizinhança 1-passo do processador P_i, $N_i^{(1)}$, é definida como segue: o processador P_j está na vizinhança 1-passo de P_i (isto é, $P_j \in N_i^{(1)}$), se ele é adjacente a P_i, isto é, $H_{ij} = H_{ji} = 1$. P_i coleta a informação de carga na vizinhança 1-passo para calcular sua distribuição de carga, $\tilde{w}_i(k)$, como segue:

$$\tilde{w}_i(k) = \frac{C_i}{C_i + \sum_{\forall j P_j \in N_i^{(1)}} C_j} \left(w_i(k) + \sum_{\forall j P_j \in N_i^{(1)}} w_j(k)\right) \qquad \{4\}$$

O primeiro fator da multiplicação denota o poder computacional do conjunto $P_i \bigcup N_i^{(1)}$ associado com P_i e o segundo fator denota o número total de tarefas no conjunto $P_i \bigcup N_i^{(1)}$.

4 Algoritmos de Balanceamento de Carga Apresentados Segundo o Modelo Iterativo de Matriz

Nessa seção, alguns algoritmos de balanceamento de carga serão adaptados ao Modelo Iterativo de Matriz. Os procedimentos de balanceamento de carga Random, Difusão e Redistribuição Completa já foram adaptados a esse modelo em [FG96].

4.1 Random

No algoritmo Random, descrito em [ELZ86], [FG96], [LLC96] e [MD94], quando um processador P_i decide enviar uma fração α do excesso de suas tarefas para outro processador, o número de tarefas atribuídas a ele, $w_i(k)$, é maior que um certo threshold, $w_i^*(k)$. Assim, o processador P_i envia $\alpha(w_i(k) - w_i^*(k))$ tarefas, se $(w_i(k) > w_i^*(k))$. O processador de destino P_e é escolhido randomicamente, do conjunto de processadores, para o envio dessa carga.

Um elemento da matriz de participação de processadores, $A_{ij}(k)$, é igual a um, se $i = j$ e $(w_i(k) > w_i^*(k))$. Todos os outros elementos de A são zero. O elemento $M_{ie} = \alpha$ e $M_{ij} = 0$ se $j \neq e$.

Para esse algoritmo, as políticas são:

- de transferência $\rightarrow$ sobrecarga de um nó do sistema;
- de informação $\rightarrow$ nenhum nó do sistema divulga seu número de tarefas para os outros processadores;
- de localização $\rightarrow$ seleção randômica do nó de destino.

A equação {2} pode ser reescrita como:

$$w_i(k+1) = w_i(k) - \alpha A_{ii}(k)(w_i(k) - w_i^*(k)) + \sum_{j \neq i} A_{jj}(k) M_{ji}(k)(w_j(k) - w_j^*(k)) + \lambda_i(k) - \mu_i(k) \qquad \{5\}$$

4.2 Threshold ou Sender-Initiated

No algoritmo Threshold, descrito em [ELZ86], [LLC96] e [MD94], quando um processador P_i decide enviar uma fração α do excesso de suas tarefas para outro processador, o número de tarefas atribuídas a ele, $w_i(k)$, é maior que um certo threshold, $w_i^*(k)$.

O processador de destino P_e é escolhido randomicamente do conjunto de processadores. Se o número de tarefas atribuídas a P_e, $w_e(k)$, for menor que o valor de threshold, $w_e^*(k)$, transfere-se $\alpha(w_i(k) - w_i^*(k))$ tarefas para P_e. Caso contrário, repete-se o processo de seleção de outro nó.

Um elemento da matriz de participação de processadores, $A_{ij}(k)$, é igual a um, se $i = j$ e $(w_i(k) > w_i^*(k))$. Todos os outros elementos de A são zero. O elemento $M_{ie} = \alpha$ e $M_{ij} = 0$ se $j \neq e$.

Para esse algoritmo, as políticas são:

- de transferência $\rightarrow$ sobrecarga de um nó do sistema;
- de informação $\rightarrow$ o nó de destino divulga seu número de tarefas para o processador que o selecionou [LLC96];
- de localização $\rightarrow$ seleção randômica do nó de destino.

A equação de distribuição de tarefas para esse algoritmo de balanceamento de carga é idêntica à equação $\{5\}$.

4.3 Shortest

No algoritmo Shortest, descrito em [ELZ86] e [LLC96], quando um processador P_i decide enviar uma fração α do excesso de suas tarefas para outro processador, o número de tarefas atribuídas a ele, $w_i(k)$, é maior que um certo threshold, $w_i^*(k)$.

Um conjunto de processadores é escolhido randomicamente do sistema. Desse conjunto, obtém-se o processador, P_e, com o menor número de tarefas atribuídas a ele. Se o número de tarefas atribuídas a P_e, $w_e(k)$, for menor que o valor de threshold, $w_e^*(k)$, transfere-se $\alpha(w_i(k) - w_i^*(k))$ tarefas para P_e. Caso contrário, repete-se o processo de seleção de um conjunto de nós.

Um elemento da matriz de participação de processadores, $A_{ij}(k)$, é igual a um, se $i = j$ e $(w_i(k) > w_i^*(k))$. Todos os outros elementos de A são zero. O elemento $M_{ie} = \alpha$ e $M_{ij} = 0$ se $j \neq e$.

Para esse algoritmo, as políticas são:

- de transferência $\rightarrow$ sobrecarga de um nó do sistema;
- de informação $\rightarrow$ o nó de destino divulga seu número de tarefas para o processador que o selecionou [LLC96];
- de localização $\rightarrow$ seleção do nó com o menor número de tarefas, a partir de um conjunto de nós selecionados randomicamente.

A equação de distribuição de tarefas para esse algoritmo de balanceamento de carga é idêntica à equação $\{5\}$.

4.4 Receiver-Initiated

No algoritmo Receiver-Initiated, descrito em [LLC96] e [MD94], quando um processador P_i decide obter uma fração α do excesso das tarefas de outro processador, o número de tarefas atribuídas a ele, $w_i(k)$, é menor que um certo threshold, $w_i^*(k)$.

O processador P_e é escolhido randomicamente do conjunto de processadores. Se o número de tarefas atribuídas a P_e, $w_e(k)$, for maior que o valor de threshold, $w_e^*(k)$, transfere-se $\alpha(w_e(k) - w_e^*(k))$ tarefas para P_i. Caso contrário, repete-se o processo de seleção de outro nó.

Um elemento da matriz de participação de processadores, $A_{ij}(k)$, é igual a um, se $i = j$ e $(w_i(k) > w_i^*(k))$. Todos os outros elementos de A são zero. O elemento $M_{ei} = \alpha$ e $M_{ej} = 0$ se $j \neq i$.

Para esse algoritmo, as políticas são:

- de transferência $\rightarrow$ subcarga de um nó do sistema;
- de informação $\rightarrow$ o nó de destino divulga seu número de tarefas para o processador que o selecionou [LLC96];
- de localização $\rightarrow$ seleção randômica do nó de destino.

A equação de distribuição de tarefas para esse algoritmo de balanceamento de carga é idêntica à equação $\{5\}$.

4.5 Simétrico

O algoritmo Simétrico, descrito em [MD94], é uma combinação das algoritmos Sender-Initiated e Receiver-Iniated. A estratégia de balanceamento de carga é ajustada de acordo com o número de tarefas atribuídas ao nó, permitindo que esse algoritmo escolha automaticamente entre o algoritmo Sender-Initiated, quando o número de tarefas atribuídas ao nó ultrapassa o valor de threshold ou o algoritmo Receiver-Initiated, quando o número de tarefas atribuídas ao nó é menor que o valor de threshold.

Os três estados apresentados a seguir são mutuamente excludentes, isto é, um nó do sistema só pode estar em um dos estados citados abaixo:

- sobrecarregado – o número de tarefas atribuídas a ele, $w_i(k)$, é maior que um certo threshold, $w_i^*(k)$
  o processador P_i decide enviar uma fração α do excesso de suas tarefas para outro processador.
  O processador de destino P_e é escolhido randomicamente do conjunto de processadores. Se o número de tarefas atribuídas a P_e, $w_e(k)$, for menor que o valor de threshold, $w_e^*(k)$, transfere-se $\alpha(w_i(k) - w_i^*(k))$ tarefas para P_e. Caso contrário, repete-se o processo de seleção de outro nó.
  Nesse caso, um elemento da matriz de migração M_{ie} é igual a α e M_{ij} é igual a zero, se $j \neq e$.
  Para esse estado, a política de transferência é a sobrecarga de um nó do sistema.

- subcarregado – o número de tarefas atribuídas a ele, $w_i(k)$, é menor que um certo threshold, $w_i^*(k)$
  o processador P_i decide obter uma fração α do excesso das tarefas de outro processador.
  O processador P_e é escolhido randomicamente do conjunto de processadores. Se o número de tarefas atribuídas a P_e, $w_e(k)$, for maior que o valor de threshold, $w_e^*(k)$, transfere-se

$\alpha(w_e(k) - w_e^*(k))$ tarefas para P_i. Caso contrário, repete-se o processo de seleção de outro nó.
Nesse caso, um elemento da matriz de migração M_{ei} é igual a α e M_{ej} é igual a zero, se $j \neq i$.
Para esse estado, a política de transferência é a subcarga de um nó do sistema.

- balanceado – o número de tarefas atribuídas a ele, $w_i(k)$, é igual a um certo threshold, $w_i^*(k)$
  o processador P_i não participa do balanceamento de carga.
  Nesse caso, $M_{ij} = 0$, para qualquer j.
  Para esse estado, a política de transferência determina que nenhuma tarefa será transferida desse nó.

Para os três casos, um elemento da matriz de participação de processadores, $A_{ij}(k)$, é igual a um, se $i = j$ e $(w_i(k) > w_i^*(k))$. Todos os outros elementos de A são zero.

A política de informação, para esse algoritmo, é a divulgação do número de tarefas do nó de destino para o processador que o selecionou. A política de localização é a seleção randômica do nó de destino.

A equação de distribuição de tarefas para esse algoritmo de balanceamento de carga é idêntica à equação {5}.

4.6 APAL

No algoritmo APAL, descrito em [LLC96], quando um processador P_i decide enviar uma fração α do excesso de suas tarefas para outro processador, o número de tarefas atribuídas a ele, $w_i(k)$, é maior que um certo threshold, $w_i^*(k)$.

O processador P_i faz uma previsão dos nós sobrecarregados e subcarregados do sistema e, então elege um processador, P_e, do conjunto de processadores.

O procedimento de previsão de estado tem a seguinte estrutura: na iteração k, cada processador participante do sistema divulga seu número de tarefas para os outros componentes. Cada nó, então, armazena essas informações e as usa, na próxima iteração $(k+1)$, para estimar a carga dos outros processadores [LLC96].

Se o número de tarefas atribuídas a P_e, $w_e(k)$, for menor que o valor de threshold, $w_e^*(k)$, transfere-se $\alpha(w_i(k) - w_i^*(k))$ tarefas para P_e. Caso contrário, repete-se o processo de eleição de outro nó, até que o número de eleições ultrapasse a um valor de eleição pré-definido, E. Se o número de eleições for maior que E, esse algoritmo para até que um nó torne-se subcarregado.

Um elemento da matriz de participação de processadores, $A_{ij}(k)$, é igual a um, se $i = j$ e $(w_i(k) > w_i^*(k))$. Todos os outros elementos de A são zero. O elemento $M_{ie} = \alpha$ e $M_{ij} = 0$ se $j \neq e$.

Para esse algoritmo, as políticas são:

- de transferência $\rightarrow$ sobrecarga de um nó do sistema;

- de informação $\rightarrow$ cada processador participante do sistema divulga o seu número de tarefas para os outros participantes;

- de localização $\rightarrow$ eleição do nó de destino a partir de uma previsão dos nós subcarregados e sobrecarregados do sistema.

A equação de distribuição de tarefas para esse algoritmo de balanceamento de carga é idêntica à equação {5}.

4.7 Redistribuição Completa

No algoritmo Redistribuição Completa, descrito em [FG96], quando a carga em qualquer processador i, $w_i(k)$, ultrapassa o valor de threshold, $w_i^*(k)$, esse algoritmo agrupa todas as tarefas do sistema e as distribui, igualmente, entre todos os nós desse sistema. Portanto, todos os processadores participam do balanceamento de carga.

Os processadores usam um algoritmo de otimização para calcular a matriz $M(k)$. O cálculo dessa matriz resulta em uma distribuição balanceada das tarefas. Por exemplo, todas as tarefas são agrupadas em um processador mestre e, então, redistribuídas igualmente usando um algoritmo ótimo de alocação de tarefas.

Pelo fato de todos os processadores participarem do balanceamento de carga, a matriz de participação do processador, $A(k)$, será uma matriz com os elementos da diagonal iguais a um e os outros elementos iguais a zero.

Para esse algoritmo, as políticas são:

- de transferência $\rightarrow$ sobrecarga de um nó do sistema;
- de informação $\rightarrow$ nenhum nó do sistema divulga seu número de tarefas para os outros processadores;
- de localização $\rightarrow$ todos os nós do sistema.

A equação de distribuição de tarefas para esse algoritmo de balanceamento de carga é idêntica à equação $\{5\}$.

4.8 Difusão

No algoritmo Difusão, descrito em [FG96], quando a carga em qualquer processador i, $w_i(k)$, ultrapassa o valor de threshold, $w_i^*(k)$, esse algoritmo decide enviar uma fração α do excesso de suas tarefas para seus vizinhos, examinando, antes, a distribuição de tarefas em cada processador vizinho.

Um elemento da matriz de participação de processadores, $A_{ij}(k)$, é igual a um, se $i = j$ e $(w_i(k) > w_i^*(k))$. Todos os outros elementos de A são zero.

Os destinos das tarefas são calculados baseados na distribuição de tarefas nos processadores vizinhos. P_i envia tarefas para P_j, se o número de tarefas em P_j, $w_j(k)$, é menor que a carga ideal em P_j, $\bar{w}_j(k)$, isto é, P_i envia $\alpha(\bar{w}_j(k) - w_j(k))$ tarefas para P_j. A matriz $M(k)$ é calculada da seguinte maneira:

(1) $M_{ij}(k) = \alpha[\bar{w}_j(k) - w_j(k)]$;

(2) $M_{ij}(k) = 1$.

O primeiro item fornece um sistema de n equações, os elementos $M_{ij}(k)$ para cada j, e $n+1$ incógnitas, todos os elementos $M_{ij}(k)$ e a constante α. O segundo item indica que a soma de todos os elementos da linha i deve ser 1, o que representa o deslocamento de todo o excesso de tarefas do processador P_i para os seus vizinhos. Como se tem um sistema de $n+1$ equações, os ítens (1) e (2), e $n+1$ incógnitas, define-se cada um dos elementos da linha i da matriz $M(k)$.

Para esse algoritmo, as políticas são:

- de transferência $\rightarrow$ sobrecarga de um nó do sistema;
- de informação $\rightarrow$ cada processador participante do sistema divulga o seu número de tarefas para os seus vizinhos;

- de localização → todos os processadores vizinhos.

A equação de distribuição de tarefas para esse algoritmo de balanceamento de carga é idêntica à equação {5}.

5 Considerações Finais

Os algoritmos de balanceamento de carga adaptativos que têm sido examinados na literatura [ELZ86][FG96][LLC96][MD94] coletam informações de estado do sistema e, baseados nessas informações, procuram fazer a melhor redistribuição da execução das tarefas de uma aplicação [ELZ86].

Claramente, a complexidade dessa coleta de informações introduz alguns pontos interessantes:

1. o efeito do *overhead* na coleta das informações → o uso de um algoritmo de balanceamento de carga adaptativo depende criticamente do *overhead* requerido para coletar as informações de estado do sistema. Esse *overhead* pode influir diretamente no tempo de execução de uma aplicação submetida ao procedimento de balanceamento. Assim, excessivos *overheads* podem anular os benefícios de uma redistribuição da execução das tarefas de uma aplicação;

2. o efeito de uma redistribuição das tarefas de uma aplicação → algoritmos de balanceamento de carga complexos contam com informações detalhadas a respeito do estado do sistema. Além desse tipo de informação demorar mais tempo para ser coletada, ela não consegue determinar a sobrecarga ou subcarga de um nó num futuro próximo. Devido a isso, uma redistribuição das tarefas de uma aplicação feita por um algoritmo de balanceamento de carga complexo que supõe estar próxima da redistribuição ótima, pode ser, de fato, muito ruim.

Baseados nos pontos levantados anteriormente, surge a seguinte questão com relação aos algoritmos de balanceamento de carga adaptativos: qual é o nível de complexidade desses algoritmos? Em [ELZ86][MD94][FG96] foi mostrado que:

- algoritmos de balanceamento de carga adaptativos simples – algoritmos com pouca ou nenhuma coleta de informações de estado, como Threshold e Random, respectivamente – produzem uma melhoria no desempenho em relação ao caso onde não há a redistribuição das tarefas;

- algoritmos de balanceamento de carga adaptativos simples têm o desempenho melhor que algoritmos de balanceamento de carga que coletam uma grande quantidade de informação de estado do sistema e que tentam fazer a "melhor" redistribuição da execução, como Shortest e Difusão;

- esses resultados são válidos sobre um grande intervalo de parâmetros do sistema.

Em [ELZ86] conclui-se que algoritmos de balanceamento de carga adaptativos simples são mais interessantes do ponto de vista prático e, de modo geral, não existem evidências de que custos de coletar e usar um volume grande de informações de estado tragam algum benefício real. Entretanto, muitos algoritmos de balanceamento de carga adaptativos complexos têm sido propostos na literatura [MD94][LLC96], visando o mesmo tempo de execução dos algoritmos simples e, além disso, uma melhor redistribuição das tarefas de uma aplicação.

Bibliografia

[BF96] Bernard, G., Folliot, B., *Caractéristiques Générales du Placement Dynamique: Syntèse et Problématique*, Ecole Placement Dynamique et Répartition de Charge, 1996.

[Chah80] Chahon, J. A., *Modelos Analíticos e de Simulação para Sistemas de Filas com e sem Prioridade*, Dissertação de Mestrado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, Outubro de 1980.

[Cybe89] Cybenko, G., *Dynamic Load Balancing for Distributed Memory Multiprocessors*, Journal Parallel Distributed Computing 7(2):279-301, 1989.

[ELZ86] Eager, D. L., Lazowska, E. D., Zahorjan, J., *Adaptive Load Sharing in Homogeneous Distributed Systems*, IEEE Transactions on Software Engineering, SE-12(5):662-675, 1986.

[FG96] Franklin, M. A., Govindan, V., *A General Matrix Iterative Model for Dynamic Load Balancing*, Parallel Computing 22(1996) 969-989

[LLC96] Lee, G., Lee, H., Cho, J., *A Prediction-based Adaptive Location Policy for Distributed Load Balancing*, Journal of Systems Architecture, 42:1-18, 1996.

[MD94] Mahieddine, K. B., Dew, P. M., *A Periodic Symmetrically Initiated Load Balancing Algorithm for Distributed Systems*, Operating Systems Review, 28(1):66-77, 1994.