Uma Taxonomia de Algoritmos de Balanceamento de Carga para Aplicações SPMD

Alexandre Plastino
Departamento de Ciência da Computação
Universidade Federal Fluminense - Niterói - RJ
plastino@dcc.uff.br

Celso C. Ribeiro
celso@inf.puc-rio.br

Noemi Rodriguez
noemi@inf.puc-rio.br

PUC-RioInf.MCC49/98
Dezembro 1998

Abstract - We propose a taxonomy of load balancing strategies for SPMD (Single Program, Multiple Data) parallel applications. This taxonomy leads to a classification of load balancing algorithms designed for data parallel applications in MIMD architectures. This classification allows more thorough comparison studies of different load balancing techniques, helping application programmers in the task of identifying the most suitable strategy for some given SPMD application under development.

Keywords: Load Balancing, SPMD, Taxonomy, Parallel Applications.

Resumo - Propõe-se uma taxonomia de algoritmos de balanceamento de carga para aplicações paralelas SPMD (Single Program, Multiple Data). Esta taxonomia viabiliza a classificação dos algoritmos de balanceamento de carga projetados para aplicações com paralelismo de dados em arquiteturas MIMD. Esta classificação facilita a comparação mais criteriosa de estratégias, auxiliando projetistas de aplicações SPMD na tarefa de identificação de estratégias de balanceamento de carga adequadas à aplicação em desenvolvimento.

Palavras-chave: Balanceamento de Carga, SPMD, Taxonomia, Aplicações Paralelas.

1 Introdução

A evolução dos sistemas paralelos tem evidenciado que o modelo MIMD (Multiple Instruction Stream, Multiple Data Stream) tem sido o mais adotado no projeto de máquinas paralelas [Duc96, Mat96, Zom96]. Neste modelo, de propósito mais geral que os demais, um computador paralelo é caracterizado por um conjunto de processadores que executam, de forma independente e assíncrona, seqüências de instruções (processos) diferentes sobre conjuntos de dados distintos.

Os computadores paralelos compostos por estações de trabalho ligadas por redes de interconexão de alta velocidade e até mesmo os ambientes distribuídos compostos por redes de estações de trabalho são considerados sistemas MIMD. Estas arquiteturas têm apresentado uma relação entre custo e desempenho bastante competitiva comparada a outros ambientes paralelos [BS96].

Os algoritmos paralelos desenvolvidos em ambientes MIMD são projetados segundo dois modelos de programação paralela: paralelismo de função (ou de controle) e paralelismo de dados. No primeiro modelo, um programa paralelo é composto por um conjunto de tarefas diferentes, que implementam funções distintas, executadas concorrentemente nos diversos processadores. Este modelo é adequado a arquiteturas que permitem a execução concorrente de diferentes instruções nos diversos processadores, como é o caso das arquiteturas MIMD.

No modelo chamado paralelismo de dados, um programa paralelo é caracterizado por um mesmo processo executado em todos os processadores sobre diferentes conjuntos de dados. Portanto, o que define uma tarefa neste modelo é a partição dos dados sobre a qual o código comum deve ser executado. Este modelo, quando associado a arquiteturas MIMD, é conhecido como SPMD (Single Program, Multiple Data Stream), no qual o mesmo programa é executado de forma assíncrona nos diferentes processadores, manipulando conjuntos distintos de dados [Qui94]. O modelo SPMD tem sido amplamente utilizado devido à facilidade de desenvolvimento e controle de um único código executado nos diversos processadores. Além disso, um número considerável de problemas pode ser resolvido por algoritmos paralelos baseados neste modelo [Mat96].

A redução do tempo de resposta de programas paralelos tem sido alvo de um grande esforço de pesquisa. O tempo de execução de um programa paralelo é definido pelo intervalo entre o instante em que o primeiro processador inicia a sua parte do processamento e o momento em que o último encerra a sua parte. A adequada distribuição de tarefas nos diversos processadores é fundamental para a minimização do tempo de execução do programa paralelo e maximização da utilização dos recursos computacionais disponíveis. A identificação desta distribuição ideal de tarefas caracteriza o problema de **balanceamento de carga**.

Um algoritmo de balanceamento de carga de uma aplicação paralela tem como objetivo, então, definir a alocação adequada aos processadores das tarefas que compõem a aplicação, de forma a tentar minimizar o seu tempo de execução. Para tanto, o algoritmo de balanceamento de carga deve controlar os diferentes fatores de desequilíbrio existentes.

Um destes fatores é a heterogeneidade da arquitetura paralela adotada. Em um ambiente paralelo com processadores heterogêneos, por exemplo, as diferentes capacidades de processamento dos diversos processadores devem ser consideradas no momento da distribuição das tarefas. Processadores com uma maior capacidade devem receber mais tarefas e aqueles com inferior poder de processamento, uma carga menor. Este fator de desequilíbrio é considerado estático, pois não varia ao longo da execução da aplicação e portanto pode ser tratado antes do início desta. Entretanto, há varios outros fatores que impedem que o problema do equilíbrio de carga seja resolvido estaticamente. Diante destes outros fatores, tornam-se necessárias estratégias dinâmicas de balanceamento de carga que adotem políticas de transferência de tarefas ou de distribuição destas ao longo da execução da aplicação. Estas políticas visam manter a quantidade de trabalho equilibrada nos diversos processadores, reduzindo os efeitos destes fatores de desequilíbrio que devem ser tratados dinamicamente.

Apresentam-se, a seguir, alguns destes conhecidos fatores dinâmicos de desequilíbrio. Em determinadas aplicações, não se conhece, previamente, a quantidade de trabalho envolvida em cada tarefa. Isto impede que se faça, antecipadamente, uma distribuição equilibrada de trabalho entre os processadores. A criação dinâmica de tarefas que ocorre em algumas aplicações, durante a sua execução, também contribui para a impossibilidade de se definir estaticamente o balanceamento de carga. A migração de tarefas que ocorre em alguns tipos de aplicação é outro fator de desequilíbrio. Destaca-se, ainda, a perda do equilíbrio da carga de trabalho durante a execução de uma aplicação paralela causada pela heterogeneidade e pela variação da carga externa à aplicação nos diversos processadores de uma máquina paralela. Para ilustrar este último caso, considere um ambiente paralelo com memória distribuída composto por um conjunto de estações de trabalho ligadas por uma rede de interconexão de alta velocidade. Neste tipo de ambiente, a quantidade de processos dos diversos usuários não é a mesma nas diferentes estações e varia ao longo do tempo. Por este motivo, são denominados ambientes não dedicados [ZLP97], adaptativos e não uniformes [KR97] ou instáveis [AM96]. Em um ambiente não dedicado, então, uma distribuição homogênea das tarefas pode provocar um desequilíbrio nos tempos de encerramento das tarefas nos diversos processadores. As estações com uma intensa carga externa à aplicação encerram a execução da sua parte das tarefas em tempos consideravelmente maiores do que os tempos das estações menos carregadas. Neste caso, portanto, estaticamente, não se consegue prever a distribuição ideal das tarefas de uma aplicação paralela de modo a equilibrar os tempos finais de execução de cada processador da máquina paralela. Fica evidente, então, a importância de se obter, dinamicamente, o equilíbrio da carga de aplicações paralelas neste tipo de ambiente.

Encontra-se em [SHK95], uma coleção de trabalhos que discutem diferentes aspectos do problema do balanceamento de carga. Uma estratégia de balanceamento de carga pode ser empregada em diferentes contextos. O primeiro caso trata-se de manter a carga equilibrada em todos os nós de um ambiente de programação distribuída ou paralela. O balanceamento de carga é executado pelo escalonador de processos, que determina em que nós do sistema os processos que se iniciam devem ser executados e, além disso, coordena uma possível política de trasferência de processos dos nós sobrecarregados para os menos ocupados [BMD94, Cyb89, ELZ86, LLC96, SKS92].

No segundo contexto, o objetivo é equilibrar a carga de trabalho de uma aplicação paralela nos diferentes processadores. Neste caso, existe ainda uma subdivisão caracterizada pelo modelo de paralelismo da aplicação: de função ou de dados. Quando se trata de paralelismo de função, o escalonador de tarefas é o responsável por distribuir, de forma equilibrada, as várias tarefas que compõem a aplicação paralela entre os processadores, obedecendo às restrições de precedência existentes entre estas tarefas. Em [CK88], foi definida uma taxonomia que classifica os diversos tipos de escalonadores.

No caso de paralelismo de dados, a estratégia de balanceamento de carga, na maioria das vezes, aparece inserida na própria aplicação, o que exige do projetista uma preocupação com a escolha da estratégia de balanceamento adequada. Em [AM96, dBS97, Eri96, FG96, FTI90, GZ97, KGGK94, KR97, LPR97, WLR93, ZLP97], apresentam-se algoritmos de balanceamento dinâmico de carga utilizados em algumas aplicações. Se por um lado, o modelo paralelismo de dados tem atraído a atenção de muitos projetistas e pesquisadores por oferecer uma estrutura mais simples e clara de programação paralela, por outro, exige um esforço na identificação do algoritmo de balanceamento de carga ideal à aplicação sendo desenvolvida.

Procurando auxiliar o desenvolvimento de algoritmos paralelos segundo o modelo paralelismo de dados, no sentido de facilitar a identificação de uma estratégia adequada de balanceamento de carga a uma determinada aplicação, o objetivo deste trabalho é desenvolver um estudo sobre estratégias de balanceamento de carga para aplicações paralelas SPMD em arquiteturas MIMD com memória distribuída.

A contribuição central deste trabalho é uma taxonomia de algoritmos de balanceamento de carga definidos para aplicações da classe SPMD. A conhecida taxonomia sobre escalonamento em sistemas distribuídos, apresentada no trabalho [CK88], trata mais objetivamente dos escalonadores de processos em sistemas distribuídos e dos escalonadores de tarefas de aplicações com paralelismo de controle, não abordando diretamente a questão do balanceamento de carga de aplicações SPMD.

Este texto encontra-se organizado da seguinte forma. Na Seção 2, define-se mais detalhadamente a classe SPMD de aplicações paralelas. Na Seção 3, apresenta-se a taxonomia de algoritmos de balanceamento de carga, contribuição principal deste trabalho. Algumas estratégias de balanceamento de carga são classificadas, na Seção 4, segundo a taxonomia proposta. A Seção 5 encerra este trabalho com algumas considerações.

2 Aplicações Paralelas SPMD

De acordo com [Qui94], a classe de aplicações paralelas SPMD (Single Program, Multiple Data Stream) é definida por aplicações projetadas segundo o modelo paralelismo de dados para serem executadas em arquiteturas MIMD (Multiple Instruction Stream, Multiple Data Stream). Desta forma, em uma aplicação SPMD, o mesmo código é executado, sem sincronismo no nível de instrução, sobre diferentes partições dos dados nos diversos processadores. A tarefa de uma aplicação SPMD é, portanto, caracterizada por uma partição dos dados sobre a qual será executado o código comum.

Uma aplicação SPMD pode ser categorizada segundo a comunicação existente entre os processadores ao longo da sua execução: as aplicações que não se comunicam, as que se comunicam em pontos de sincronismo bem definidos e as que não apresentam regularidade na comunicação. Analisa-se, nas subseções seguintes, cada uma destas categorias.

2.1 Aplicações SPMD sem Comunicação

Nesta categoria, os processadores não se comunicam ao longo da execução das tarefas que compõem a aplicação. A execução de uma tarefa não depende de dados resultantes da execução das demais. Portanto, as tarefas são completamente independentes umas das outras e podem ser executadas em qualquer processador e em qualquer ordem.

Em [Fos94], problemas que são resolvidos por estratégias deste tipo são chamados problemas embaraçosamente paralelos (*embarrassingly parallel problems*). Segundo [Mat96], este tipo de estratégia é muito comum devido à sua facilidade de codificação e à sua aplicabilidade em vários problemas. Sempre que um problema pode ser resolvido através de um *loop* de iterações independentes, uma aplicação SPMD sem comunicação pode ser adotada.

Em [Han95], estudam-se diversos paradigmas de programação paralela. Um deles, chamado paradigma de Monte Carlo, é caracterizado por vários processos idênticos, entre os quais não existe dependência, definidos a partir de condições iniciais geradas randomicamente. Toda estratégia paralela que segue este paradigma também é uma aplicação SPMD sem comunicação.

A multiplicação paralela de matrizes é um exemplo simples de problema que pode ser resolvido por meio de uma aplicação SPMD sem comunicação. Em uma possível implementação, cada tarefa se resume ao cálculo de um elemento da matriz resultante e, portanto, é definida por uma linha da primeira matriz e uma coluna da segunda. O código comum, executado em todos os processadores, calcula então o somatório dos produtos dos elementos de uma linha pelos elementos correspondentes de uma coluna.

Uma possível paralelização da metaheurística GRASP (Greed Randomized Adaptive Search Procedures) [PPMR95] para problemas de otimização combinatória consiste em executar em paralelo as sua diversas e independentes iterações. Cada iteração é executada a partir dos dados do problema e de valores gerados aleatoriamente, obtendo um ótimo local. Ao final da aplicação, a melhor solução para o problema em questão é selecionada. A execução independente das iterações caracteriza uma aplicação SPMD sem comunicação.

Muitas aplicações em computação gráfica e processamento de imagem também pertencem a essa categoria [Eri96, LRN97, dBS97]. Muitos problemas desta área se caracterizam pela necessidade de execução de um mesmo processamento sobre os diversos *pixels* que formam uma imagem. O processamento de cada unidade da imagem, nestes problemas, não depende do resultado do processamento das demais. As estratégias paralelas podem ser formadas, então, por um código comum executado sobre diferentes partes da imagem, sem dependência entre estas tarefas. Portanto, pertencem também à classe de aplicações SPMD sem comunicação.

Há também, na área de banco de dados, aplicações paralelas desta classe. Em [CCH93], são analisadas estratégias paralelas para o problema do cálculo do fecho transitivo de relações binárias. Em uma delas, o cálculo dos sucessores de um conjunto de nós é executado particionando-o e executando o mesmo procedimento, sem comunicação, sobre cada elemento resultante do particionamento. Outro problema na área de processamento de consultas, o cálculo paralelo da junção relacional, também admite solução SPMD sem comunicação [LOT94].

A execução de uma aplicação paralela SPMD sem comunicação, representada genericamente pela Figura 1, se divide em três fases. Na primeira fase (D), executa-se a distribuição inicial das tarefas entre os processadores. Na Figura 1, representa-se uma distribuição de dez tarefas para cada processador disponível. Na segunda fase (E), são executadas as tarefas que compõem a aplicação. Cada pequeno retângulo representa a execução de uma tarefa em um determinado processador ao longo de certo tempo de execução. A última fase (C) representa a coleta e análise dos resultados de cada processador. Tanto a distribuição quanto a coleta dos resultados são fases comuns à maioria das aplicações SPMD e são realizadas por um dos processadores alocados para a execução da aplicação.

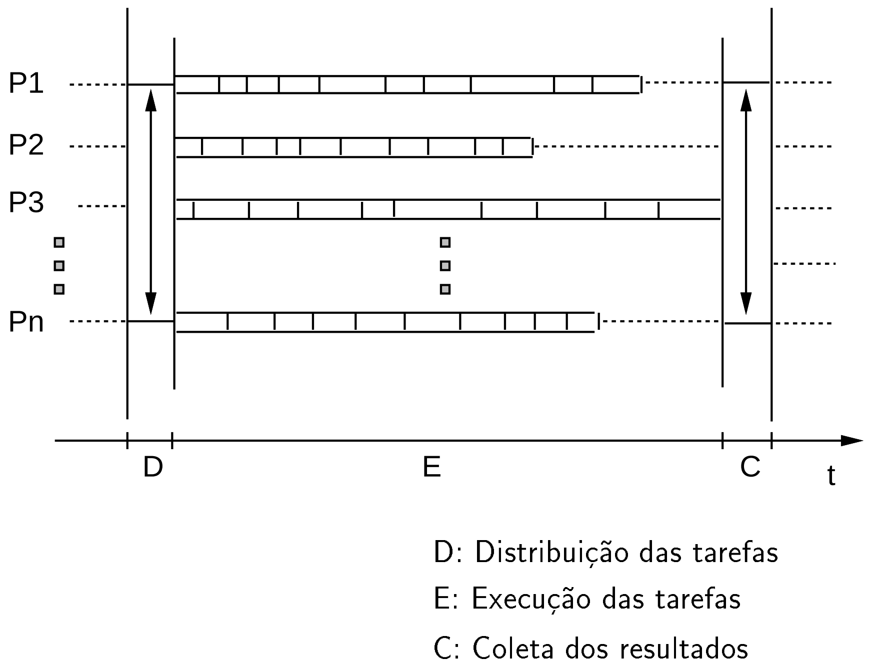

Figura 1: Execução de uma aplicação SPMD sem comunicação.

A Figura 1 ilustra a ocorrência de um desequilíbrio dos tempos finais dos processadores apesar da distribuição homogênea de tarefas entre estes. Para evitar o desequilíbrio ilustrado, utilizam-se algoritmos de balanceamento de carga. Estes têm por objetivo minimizar o tempo de execução de uma aplicação paralela através da adoção de políticas de distribuição e transferência de tarefas, adequadas à aplicação em questão.

As aplicações SPMD sem comunicação possuem características que facilitam a utilização de estratégias de balanceamento de carga. Devido à independência entre as tarefas destas aplicações, as tarefas não realizadas podem ser, a qualquer momento, convenientemente transferidas de um processador a outro com o objetivo de equilibrar a carga da aplicação. Cabe estabelecer que a carga de uma aplicação SPMD sem comunicação, em um determinado processador, em um instante qualquer, é considerada como o conjunto de tarefas ainda a serem executadas neste processador.

2.2 Aplicações SPMD com Comunicação Síncrona

Nesta categoria, a comunicação entre os processadores é realizada em pontos de sincronismo bem definidos. Nestas sincronizações, todos os processadores se voltam para a troca de informações. As tarefas que compõem aplicações desta categoria são caracterizadas pela necessidade de troca periódica de informações com as demais. A execução das tarefas entre dois pontos de sincronismo corresponde à execução de uma iteração da aplicação. A execução de uma iteração depende dos resultados da anterior. As aplicações SPMD com comunicação síncrona podem ser também denominadas aplicações paralelas iterativas.

Em uma categorização apresentada em [Mat96], uma aplicação SPMD com comunicação síncrona seria classificada como síncrona (*synchronous*) ou fracamente síncrona (*loosely synchronous*) dependendo, respectivamente, se os elementos de dados que definem as tarefas são equivalentes (isto é, se a demana computacional das diferentes tarefas é a mesma) ou não.

A Figura 2 ilustra, genericamente, a execução das fases que compõem uma aplicação SPMD com comunicação síncrona. Em uma fase inicial (D), ocorre a distribuição das tarefas entre os processadores. Em seguida, é executada uma seqüência de iterações. Cada iteração corresponde à excecução das tarefas entre dois pontos de sincronismo (S), nos quais estas tarefas trocam as informações necessárias à continuidade do seu processamento. Na última fase (C), os resultados gerados por cada processador são coletados e analisados. Apesar da necessidade de comunicação nesta categoria, entre quaisquer dois pontos de sincronismo, não há troca de informações entre as tarefas, ou seja, não há comunicação entre os processadores.

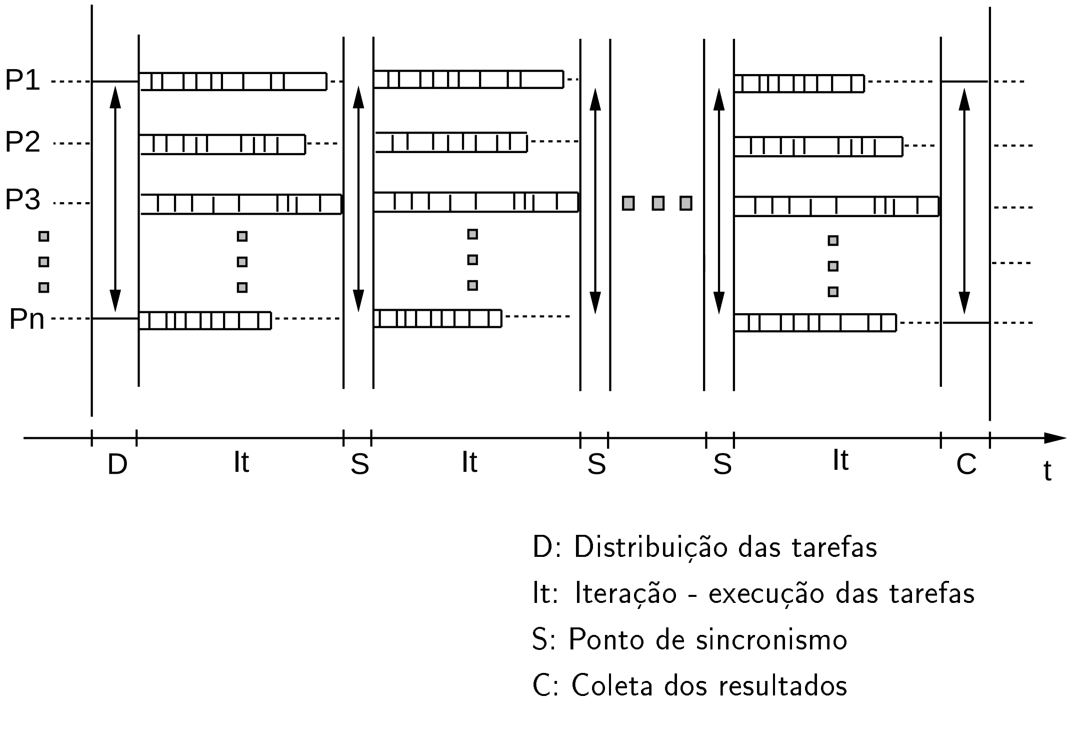

Figura 2: Execução de uma aplicação SPMD com comunicação síncrona.

O problema dos N-corpos [FG96] possui técnicas de resolução cuja paralelização exemplifica a categoria das aplicações SPMD com comunicação síncrona. Neste problema, simula-se o movimento de um conjunto de partículas causado pela força gravitacional existente entre cada par destas partículas. A simulação se desenvolve através de uma seqüência de iterações. Em cada iteração, a partir das posições correntes de todas as partículas, calculam-se as forças incidentes em cada partícula e sua nova posição. Inicialmente, distribuem-se as tarefas entre os processadores. Cada tarefa é definida pela simulação do movimento de uma partícula. Em cada iteração, é calculada a nova posição de cada partícula a partir das posições correntes de todas. Ao final de cada iteração, ou seja, em cada ponto de sincronismo, todos os processadores devem enviar as novas posições das suas partículas para todos os outros processadores. Ao final da última iteração, as posições de todas as partículas são coletadas.

Na área de matemática computacional, há diversas técnicas de solução para problemas numéricos cuja implementação paralela é naturalmente uma aplicação SPMD com comunicação síncrona. Em [BT97], destacam-se algoritmos paralelos síncronos que implementam métodos diretos e iterativos para resolução de sistemas de equações lineares e métodos iterativos para resolução de problemas não lineares.

O tempo de execução de uma aplicação paralela iterativa é definido pela soma dos tempos de cada iteração adicionado dos tempos de comunicação. Portanto, o desequilíbrio dos tempos em que os processadores encerram cada iteração, ilustrado na Figura 2, deve ser evitado.

Em [FG96], são avaliados alguns algoritmos de balanceamento de carga para uma aplicação iterativa - o problema dos N-corpos. Neste trabalho, o balanceamento de carga é executado nos pontos de sincronismo com o objetivo de definir uma redistribuição da carga para a iteração seguinte. Esta redistribuição foi utilizada devido a possíveis migrações de partículas entre os processadores, ou melhor, entre as regiões do espaço associadas aos processadores.

A execução do balanceamento de carga em uma aplicação paralela iterativa ocorre, normalmente, entre as iterações da aplicação, aproveitando o sincronismo entre os processadores.

2.3 Aplicações SPMD com Comunicação Assíncrona

Nas aplicações desta categoria, a comunicação existente entre os processadores não ocorre em pontos de sincronismo bem definidos. O padrão de comunicação desta categoria é dependente da aplicação. Muitas são caracterizadas pela comunicação de forma irregular e imprevisível. Segundo [Mat96], a utilização de aplicações SPMD com estas características é pouco comum. Isto ocorreria não por falta de problemas que poderiam se beneficiar de estratégias SPMD com comunicação assíncrona, mas pela maior complexidade na construção deste tipo de aplicação.

A paralelização da técnica *branch-and-bound* apresentada em [Fos94] ilustra esta categoria de aplicações SPMD. Esta técnica, utilizada para solucionar problemas de otimização combinatória, desenvolve-se através de uma busca em uma árvore que representa as soluções do problema em questão. Cada ramo representa uma solução. A procura pela solução ótima leva em consideração um limite (*bound*) que indica a melhor solução encontrada até o momento. Isto permite desconsiderar subárvores que, necessariamente, não levam a soluções melhores que a solução representada pelo limite e, conseqüentemente, permite reduzir a área de busca. Na paralelização de [Fos94], uma tarefa é definida por uma subárvore a ser percorrida, considerando o limite corrente. Portanto, este limite deve estar sempre atualizado em todos os processadores. Esta atualização pode ser projetada de diversos modos. Em um desses, sempre que um processador encontra uma solução melhor do que a representada pelo limite, envia uma mensagem aos demais processadores informando o novo limite. A comunicação existente, neste caso, não é previsível e nem regular, caracterizando uma aplicação SPMD com comunicação assíncrona.

A Figura 3 ilustra, genericamente, a execução desta paralelização da técnica *branch-and-bound*. Na primeira fase (D), executa-se a distribuição inicial das tarefas entre os processadores, ou seja, das sub-árvores a serem percorridas. Na segunda fase (E), são executadas as tarefas que compõem a aplicação. Estão representados quatro pontos nos quais algum processador descobre um novo limite e o informa aos outros processadores. Por exemplo, em L1, o processador P2 divulga aos demais o novo limite. Ao final da execução da aplicação (F), o limite corrente representa a solução ótima do problema em questão.

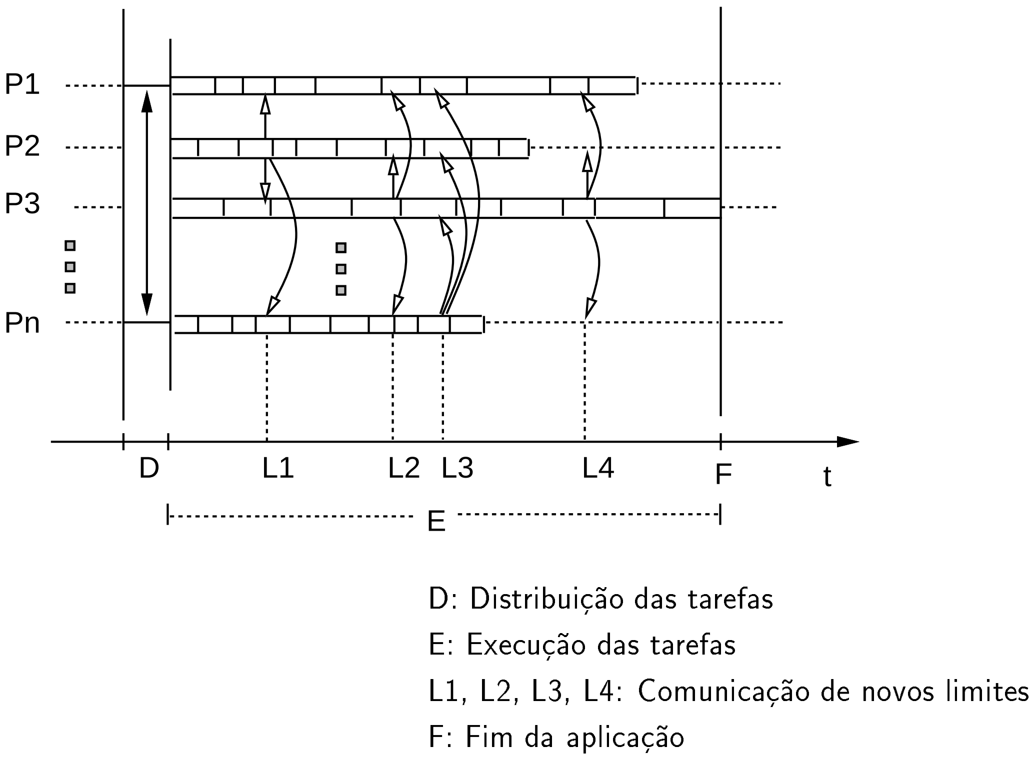

Figura 3: Execução de uma aplicação SPMD com comunicação assíncrona.

O desconhecimento prévio do tamanho da sub-árvore que será percorrida em cada tarefa é um dos fatores que impede que se realize, antecipadamente, uma distribuição equilibrada de tarefas entre os processadores. Cabe ressaltar que, apesar da existência da comunicação de novos limites entre os processadores, as tarefas desta aplicação podem ser transferidas entre estes, viabilizando a utilização de estratégias de balanceamento de carga.

Em [BT97], encontram-se algoritmos paralelos para resolver problemas lineares e não-lineares de fluxos em rede, que também ilustram a categoria das aplicações SPMD com comunicação assíncrona.

A inexistência de um padrão de comunicação para esta categoria dificulta a identificação de uma forma genérica de utilização de estratégias de balanceamento de carga. A adoção de um algoritmo de balanceamento de carga deve ser estruturada, caso a caso, de acordo com as características da aplicação paralela assíncrona sendo projetada.

3 Taxonomia

Nesta seção, propõe-se uma taxonomia de algoritmos de balanceamento de carga utilizados em aplicações com paralelismo de dados em ambientes MIMD, ou seja, aplicações SPMD. A Figura 4 contém uma representação hierárquica da taxonomia proposta. A definição desta taxonomia foi motivada pela ausência de um tratamento específico à questão do balanceamento de carga em aplicações SPMD na taxonomia apresentada em [CK88].

Serão adotadas as seguintes convenções. Cada classe desta taxonomia pode ter várias subclassificações. Cada subclassificação é representada por um número e formada por duas subclasses. Nas subseções a seguir, caracteriza-se a taxonomia proposta através da definição das suas respectivas classes. Cada subclassificação será tratada em uma subseção.

3.1 Momento da Distribuição

Toda estratégia de balanceamento de carga para aplicações SPMD possui uma **política de distribuição**. Esta se refere aos critérios que definem o particionamento dos dados entre os processadores, ou seja, critérios que definem de que forma e em que momento deve ser executada a distribuição das tarefas entre os processadores. Em função do momento em que ocorre esta distribuição, uma estratégia de balanceamento de carga pode ser classificada como estática ou dinâmica.

Uma estratégia *estática* define e realiza, antes da execução das tarefas da aplicação, a

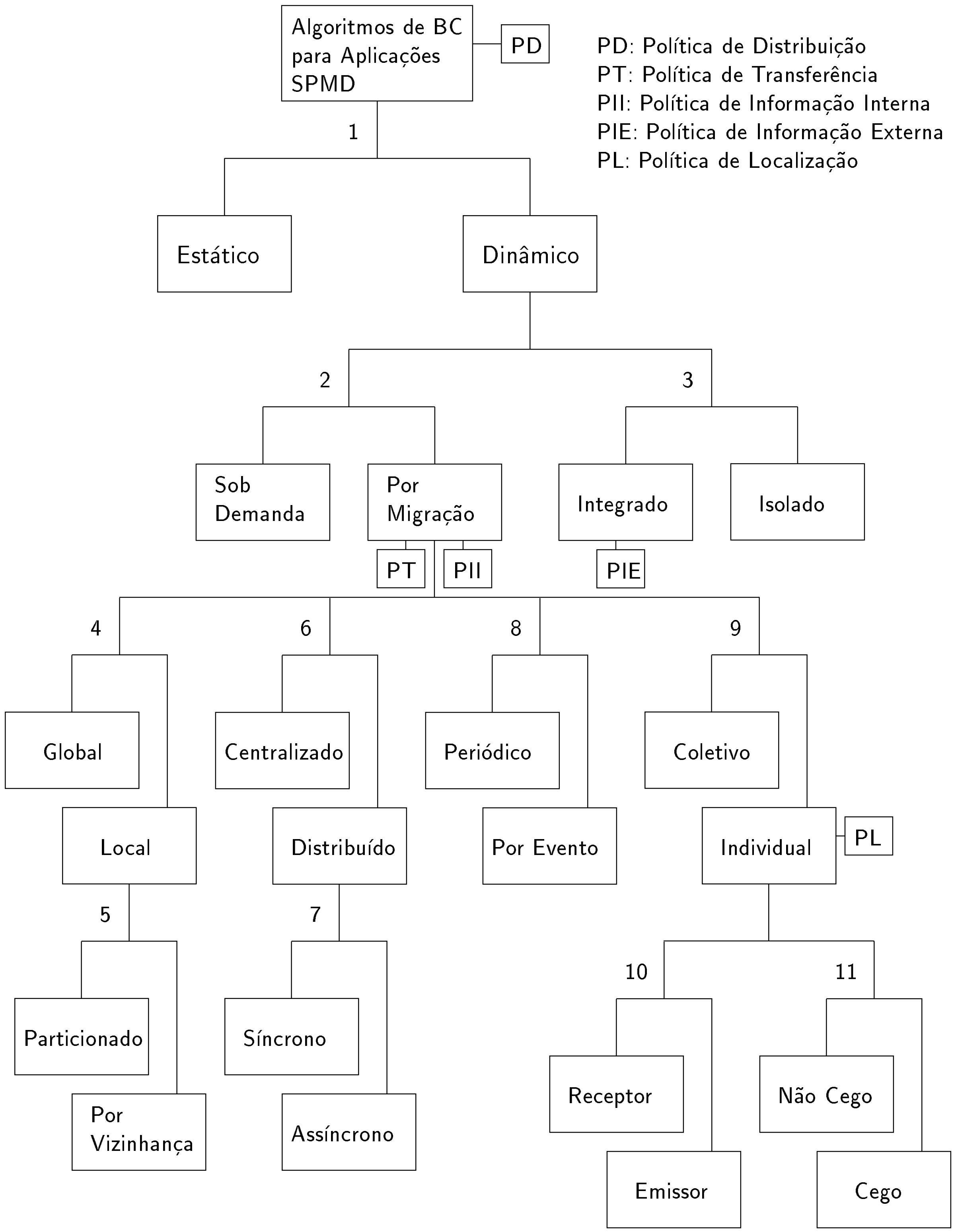

Figura 4: Taxonomia de algoritmos de balanceamento de carga para aplicações SPMD.

distribuição equilibrada destas tarefas, ou seja, o particionamento equilibrado dos dados. Este tipo de estratégia é adequado em raras situações, nas quais não existem os fatores dinâmicos de desequilíbrio. Na presença dos fatores dinâmicos de desequilíbrio, uma estratégia estática se confunde com a ausência de balanceamento de carga, pois se resume ao particionamento inicial e equilibrado dos dados e não consegue anular ou reduzir os efeitos dos fatores dinâmicos de desequilíbrio ao longo da execução da aplicação.

Uma estratégia de balanceamento de carga é *dinâmica* quando a alocação das tarefas aos processadores ocorre durante a execução da aplicação ou, alternativamente, quando ocorre no início da sua execução mas pode ser dinamicamente alterada através da transferência de tarefas. Os algoritmos de balanceamento de carga nesta categoria são capazes de controlar o equilíbrio da carga de uma aplicação diante dos diferentes fatores dinâmicos de desequilíbrio existentes em diversas aplicações e também presentes em muitos ambientes distribuídos e paralelos. São dinâmicas as estratégias encontradas em [AM96, dBS97, Eri96, FG96, FTI90, GZ97, KGGK94, KR97, LPR97, WLR93, ZLP97]. Cabe ressaltar que, para que a utilização de uma estratégia dinâmica seja satisfatória, o tempo economizado com a redução da variação dos tempos finais de execução dos diversos processadores deve ser maior do que o tempo necessário à execução do algoritmo de balanceamento de carga. Muitas vezes, permitir algum desequilíbrio, executando um algoritmo de balanceamento de carga mais simples, pode ser mais vantajoso do que atingir o equilíbrio exato através de um algoritmo muito mais caro [ELZ86, FG96, WLR93].

Em muitos trabalhos sobre balanceamento de carga, por exemplo, em [CK88, FG96, ZLP97], as estratégias já haviam sido classificadas como estáticas e dinâmicas com a mesma semântica tratada nesta taxonomia. Porém, cabe observar que, quando se trata de balanceamento de carga de sistemas distribuídos, encontram-se algumas variações em relação a esta classificação. Em [ELZ86], os algoritmos são classificados como adaptativos ou estáticos, dependendo, respectivamente, se levam ou não em consideração o estado corrente do sistema no momento em que determinam em qual nó do sistema um dado processo será executado. Em [LLC96], os algoritmos de balanceamento de carga são categorizados como estáticos, dinâmicos e adaptativos, seguindo as definições das classes de escalonadores de mesmos nomes apresentadas em [CK88]. Os estáticos, como em [ELZ86], não consideram o estado corrente do sistema. Os dinâmicos, equivalente aos adaptativos em [ELZ86], consideram este estado. E os adaptativos de [LLC96] são os que, além de considerar o estado corrente do sistema, alteram suas políticas de ação para melhor reagir às mudanças de estado do sistema.

3.2 Forma da Distribuição

Os algoritmos dinâmicos de balanceamento de carga podem ser subclassificados também de acordo com a forma de distribuição das tarefas da aplicação.

Nos algoritmos *sob demanda*, a distribuição das tarefas é feita ao longo da execução da aplicação de acordo com o comportamento dos processadores. Inicialmente, um bloco de tarefas é distribuído, por um coordenador, para cada processador. Em seguida, cada processador recebe um novo bloco de tarefas quando encerra a execução da parte recebida anteriormente. Desta forma, os processadores com uma carga externa elevada recebem, ao longo da execução da aplicação, menos tarefas, pois apresentam um desempenho inferior ao daqueles menos carregados. Ou ainda, os processadores que recebem tarefas que exigem um tempo maior de processamento acabam executando menos tarefas do que aqueles que recebem tarefas menores. Na distribuição sob demanda, não há transferência de carga entre os processadores que executam as tarefas. Há apenas a comunicação entre um coordenador, que controla a distribuição de todas as tarefas, e os demais processadores que as executam. Algoritmos sob demanda podem ser encontrados em [Eri96, FTI90, LPR97, dBS97].

A política de distribuição das estratégias desta classe podem adotar a variação do número de tarefas nos blocos que são distribuídos dinamicamente, ou seja, em algumas estratégias, a quantidade de tarefas distribuída em cada bloco (ou sua granularidade) é fixa e, em outras, varia.

Se a taxa de requisição de novos blocos de tarefas enviadas ao coordenador pelos processadores for maior que a capacidade do coordenador de atendê-las, este processamento centralizado de distribuição sob demanda pode se tornar um gargalo do algoritmo de balanceamento de carga. Portanto, a definição de um número muito grande de pequenos blocos de tarefas, assim como o aumento do número de processadores subordinados ao coordenador, pode comprometer o desempenho de uma estratégia sob demanda. O coordenador pode não atender a tempo a demanda apresentada e deixar alguns processadores ociosos por algum tempo. Conclui-se, desta forma, que estratégias desta classe podem não ser escaláveis. Em [FTI90], propõe-se uma solução para o problema da escalabilidade identificado nas estratégias sob demanda: uma distribuição de tarefas sob demanda em vários níveis. Um coordenador principal distribui sob demanda tarefas a outros coordenadores. Cada um destes distribui sob demanda as tarefas recebidas a um grupo próprio de processadores que executam as tarefas. Quando um processador conclui suas tarefas, requisita outras ao seu coordenador. Quando as tarefas de um coordenador acabam, este solicita outras ao coordenador principal. Este esquema possui apenas dois níveis, mas podem ser definidos quantos forem necessários para que nenhum coordenador, nem o principal, nem os outros, se torne um gargalo.

Um algoritmo é classificado como sendo *por migração* quando, no início da execução da aplicação, realiza a distribuição equilibrada de todas as suas tarefas entre os processadores e, durante a execução da aplicação, o balanceamento de carga é executado através da migração de tarefas de processadores sobrecarregados para os subcarregados. As estratégias desta classe se caracterizam pela **política de transferência** adotada. Esta se refere ao critério utilizado na definição das transferências de carga necessárias ao balanceamento, ou seja, na definição do processador origem, do processador destino e da respectiva carga de cada transferência. São dinâmicas e por migração as estratégias encontradas em [AM96, FG96, GZ97, KGGK94, KR97, WLR93, ZLP97].

Além da política de transferência, os algoritmos por migração são caracterizados por uma **política de informação interna**. Nesta, define-se o **índice de carga interna** a ser utilizado pela estratégia de balanceamento de carga. Este índice representa a carga de trabalho, relativa à aplicação, em cada processador. Em geral, este índice é definido pela quantidade de tarefas ainda não executadas em cada processador. A política de informação interna determina também como e com que freqüência são obtidos e atualizados os índices de carga interna nos diversos processadores.

3.3 Utilização de Índice de Carga Externa

Em muitos ambientes, a aplicação paralela cuja carga se deseja balancear concorre com diversos outros processos, paralelos ou seqüenciais, pelos recursos computacionais disponíveis. Um conjunto de estações de trabalho ligadas por uma rede de interconexão de alta velocidade é um exemplo deste tipo de ambiente. A concorrência pelos recursos ocorre em graus diferentes nos diversos processadores, além de não permanecer constante. Esta heterogeneidade e variação caracterizam um fator dinâmico de desequilíbrio que deve ser tratado pelas estratégias de balanceamento de carga. Para tanto, estas estratégias devem utilizar algum **índice de carga externa**. Este reflete a carga externa à aplicação nos diversos processadores, ou seja, a quantidade de outros processos concorrentes.

As estratégias dinâmicas de balanceamento de carga também são classificadas quanto à utilização ou não de índices de carga externa na definição das decisões que visam o equilíbrio da carga. Estas decisões são estabelecidas pela política de distribuição das estratégias dinâmicas. No caso das estratégias dinâmicas e por migração, estas decisões são definidas também pela política de transferência.

Estratégias *integradas* são aquelas que utilizam algum índice de carga externa. Procuram, ao considerar a existência de outros processos concorrentes aos recursos disponíveis, integrar-se ao ambiente e, conseqüentemente, tomar decisões mais precisas durante o balanceamento de carga.

As estratégias desta classe são caracterizadas por uma **política de informação externa**. Esta, de forma análoga à política de informação interna das estratégias por migração, define o índice de carga externa a ser utilizado, além de estabelecer a forma e a freqüência de obtenção e atualização deste índice nos diversos processadores. Um índice de carga externa, muito utilizado por estratégias que visam o balanceamento de carga de sistemas distribuídos, é representado pelo tamanho da fila de execução de cada processador do sistema [ELZ86, LLC96].

Quando se trata de balanceamento de carga de aplicações SPMD, outro tipo de índice tem sido utilizado: a capacidade de processamento demonstrada pelo processador ao realizar um conjunto de tarefas da própria aplicação. Esta capacidade pode ser representada pelo tempo que o processador leva para realizar uma certa quantidade de tarefas ou de um determinado tipo de operação. Os processadores mais lentos na execução destas tarefas são os que apresentam uma maior carga externa à aplicação. A adoção deste tipo de índice de carga é encontrada em [AM96, KR97, ZLP97]. Em [AM96, KR97], escolheu-se avaliar a capacidade do processador a partir do tempo necessário para executar um determinado tipo de tarefa. Em [ZLP97], a capacidade do processador é avaliada a partir da quantidade de tarefas da aplicação executadas em um intervalo de tempo. Cabe ressaltar que, este tipo de índice é adequado quando a quantidade de trabalho envolvida em cada tarefa for aproximadamente a mesma. Caso contrário, a execução de poucas tarefas em um determinado tempo pode indicar que o processador está com carga externa alta ou que recebeu tarefas que exigiam uma quantidade de trabalho superior às demais.

Ainda em relação à escolha de um índice de carga externa, pode-se inferir que, em aplicações nas quais a demanda computacional das tarefas é constante, a carga interna, isto é, a carga ainda a ser executada em cada processador, pode ser considerada um reflexo da carga externa em cada processador. Portanto, pode ser utilizada para avaliar a carga externa. Se a distribuição inicial de tarefas foi homogênea, uma quantidade elevada de tarefas a serem executadas em um processador pode indicar que este encontra-se mais lento e com uma carga externa maior que os demais processadores.

Estratégias *isoladas* são aquelas que não utilizam índices de carga externa ao tomar suas decisões de balanceamento de carga. São adequadas aos ambientes dedicados, nos quais não há outros processos concorrentes no sistema, ou seja, não existe a carga externa.

Deve-se considerar que a utilização de índices de carga pode tornar as decisões tomadas pela estratégia de balanceamento de carga mais precisas e eficientes. Porém, deve-se tomar cuidado com o custo de obtenção e manutenção desses índices pois, dependendo do caso, pode anular os benefícios alcançados pelas decisões mais apuradas que estes índices proporcionam.

Exemplos de algoritmos que não trabalham com índices de carga externa podem ser encontrados em [FG96, GZ97, KGGK94, LPR97, WLR93]. Em [FG96], os algoritmos de balanceamento de carga adotados na simulação paralela dos N-corpos não consideram o desequilíbrio dinâmico causado pela variação e heterogeneidade da carga externa. Neste trabalho, o único fator de desequilíbrio considerado é aquele causado pela migração de partículas, que representam tarefas, entre as áreas associadas aos diferentes processadores. A quantidade total de partículas permanece constante ao longo da execução da aplicação e deve estar equilibradamente distribuída entre os diversos processadores. O balanceamento de carga é executado apenas para ajustar este equilíbrio em função de constantes migrações destas partículas. Em [LPR97], após a apresentação dos resultados da implementação de uma estratégia sob demanda e isolada, encontra-se a sugestão de se utilizar um índice de carga externa na definição da variação da granularidade dos blocos de tarefas enviados pelo processador coordenador. Cabe ressaltar, entretanto, que uma estratégia sob demanda, mesmo isolada, indiretamente, trata da variação e heterogeneidade da carga externa. Os processadores com carga externa leve executam mais rapidamente as suas tarefas e requisitam outras, enquanto os mais carregados requisitam menos tarefas pois levam um tempo maior para processá-las. Verifica-se ainda que, neste tipo de estratégia, a variação da carga externa em um processador implica em uma variação na freqüência das requisições de novas tarefas feitas por este. Em [GZ97, KGGK94, WLR93], a carga externa à aplicação não é considerada pelos algoritmos apresentados. Estes adotam como medida de carga apenas a quantidade de tarefas ainda não executadas em cada processador, ou seja, um índice de carga interna à aplicação.

3.4 Escopo do Balanceamento de Carga

Conforme observado na Seção 3.2, as estratégias por migração se caracterizam pela política de transferência adotada, ou seja, pelos critérios utilizados na definição das transferências de carga necessárias ao balanceamento. Estas transferências de carga, em alguns casos, podem ocorrer entre quaisquer processadores, em outros, apenas entre processadores que pertençam ao mesmo grupo, de acordo com algum agrupamento dos processadores previamente estabelecido. Uma estratégia por migração pode, então, ser subcategorizada em relação a esta possível divisão dos processadores em grupos sobre os quais é executada a estratégia de balanceamento de carga em questão. Pode-se dizer ainda que esta classificação se refere ao escopo do balanceamento de carga: global ou local.

Em uma estratégia *global*, as transferências de carga podem ser executadas entre quaisquer processadores do sistema, não havendo limites definidos por grupos de processadores. Uma estratégia global pode ter acesso a informações de todos os processadores do sistema. Análises de estratégias globais são encontradas em [AM96, FG96, GZ97, KGGK94, KR97, WLR93, ZLP97].

Em uma estratégia *local*, as transferências de carga são executadas entre processadores do mesmo grupo. Estes grupos, disjuntos ou não, são previamente definidos. As decisões de uma estratégia local são tomadas, separadamente, por grupos de processadores, considerando-se apenas as informações locais a cada grupo. A utilização de estratégias locais é encontrada em [AM96, FG96, WLR93, ZLP97].

Uma estratégia global, por ter acesso a informações sobre o estado de todo o ambiente, pode atingir um equilíbrio de carga mais preciso e mais rapidamente que uma estratégia local. Esta última executa o balanceamento dentro de cada grupo, o que pode gerar um desequilíbrio entre os diferentes grupos e o equilíbrio global pode nunca ser atingido. Por outro lado, o custo de comunicação de uma estratégia global pode ser superior devido à necessidade de uma divulgação mais ampla das informações sobre todo o sistemas.

3.5 Formação dos Grupos

Os algoritmos locais podem ser subdivididos de acordo com o critério de definição da localidade, ou seja, o critério de formação dos grupos de processadores.

Em um algoritmo local *particionado*, os grupos são definidos de forma que todo processador pertença a um e somente um grupo, sendo então todos disjuntos. Desta forma, se não for implementada uma política de transferência de carga entre os grupos de processadores, pode haver desbalanceamento de carga e, conseqüentemente, o balanceamento global nunca ser atingido. Exemplos de algoritmos locais particionados são encontrados em [ZLP97].

Quando são utilizados ambientes MIMD com processadores heterogêneos, pode ser interessante que os grupos de processadores, em um algoritmo particionado, sejam definidos de modo que tenham aproximadamente a mesma capacidade computacional. Desta forma, anula-se um fator estático de desequilíbrio.

Em um algoritmo com localidade definida *por vizinhança*, os grupos são formados considerando-se a vizinhança entre os processadores na topologia física da rede de interconexão do ambiente paralelo ou na topologia lógica, definida pela aplicação. Nesta subclasse, cada processador pode pertencer a mais de um grupo. Portanto, existe a transferência de carga entre grupos e, de forma diferente da subclasse dos algoritmos particionados, o equilíbrio global pode ser atingido após algumas execuções do balanceamento de carga, possivelmente com uma convergência lenta. Em determinados algoritmos globais, o equilíbrio pode ser atingido com apenas uma execução do balanceamento, pois este pode ter acesso a informações sobre o estado de todos os processadores. Em [AM96, FG96, WLR93] são utilizados algoritmos locais por vizinhança.

3.6 Localização da Execução do Algoritmo

Como visto anteriormente, o balanceamento de carga realizado por algoritmos dinâmicos e por migração ocorre ao longo da execução da aplicação através da transferência de tarefas entre os processadores. Estes algoritmos podem ser classificados quanto ao local em que são definidas estas transferências que visam o equilíbrio de carga. Esta classificação encontra-se em [CK88], assim como em muitos outros trabalhos, com este mesmo significado.

Um algoritmo é *centralizado* se é executado em um único processador que define as transferências de carga necessárias e as informa aos processadores envolvidos nessas transferências para que estes as realizem. Este centralizador avisa aos processadores envolvidos quais são os processadores origem e destino de cada transferência e a respectiva carga a ser transferida. Estratégias centralizadas são encontradas em [AM96, GZ97, KR97, ZLP97].

Uma estratégia centralizada e local pode ter um centralizador em cada grupo de processadores ou ter um único processador central que implemente a política de transferência para todos os grupos. Em [ZLP97], a estratégia centralizada e local considerada adota a segunda alternativa.

Um algoritmo por migração é considerado *distribuído* se é executado localmente em cada processador. Para tanto, o código do algoritmo deve estar replicado em todos os processadores. Cada processador, isoladamente, de acordo com a política de transferêcia adotada, define as transferências de carga necessárias nas quais está envolvido e as realiza. Exemplos de estratégias distribuídas são encontrados em [AM96, FG96, GZ97, KGGK94, WLR93, ZLP97].

Em [ZLP97], foram utilizados quatro algoritmos de balanceamento de carga definidos a partir das possíveis combinações das categorizações global versus local e centralizado versus distribuído. Apesar do objetivo do trabalho não ser a identificação da melhor combinação, mas sim mostrar a aplicabilidade das previsões do modelo de custo proposto, os algoritmos distribuídos tiveram, para as aplicações tratadas, um desempenho superior aos centralizados e a eficiência dos algoritmos globais e locais se mostrou dependente da aplicação.

Em muitos algoritmos centralizados, o processador responsável pelo balanceamento de carga, dependendo do número de processadores do sistema, pode se tornar um gargalo. Isto, porque o processador central, dependendo da política de transferência adotada, deve receber informações de todos os outros processadores, executar o algoritmo de balanceamento de carga e devolver aos processadores as decisões que cabem a cada um. Este fato compromete a escalabilidade deste tipo de algoritmo. Se por um lado um algoritmo distribuído pode ajudar a resolver este problema, pois todos executam o algoritmo de balanceamento de carga e, portanto, definem as próprias ações, por outro, pode exigir um maior custo de comunicação quando há a necessidade de divulgar a todos os processadores as informações sobre cada um.

3.7 Sincronismo

A execução de um algoritmo distribuído, conforme visto na seção anterior, ocorre em todos os processadores. Alguns algoritmos são executados de forma síncrona, ou seja, ao mesmo tempo em todos os processadores. Em outros, a execução distribuída ocorre de forma assíncrona. Um algoritmo distribuído pode, então, ser classificado quanto à existência ou não deste sincronismo.

A execução de um algoritmo distribuído *síncrono* é caracterizada pela participação, ao mesmo tempo, de todos os processadores envolvidos no balanceamento de carga. Ao longo da execução da aplicação, há pontos de sincronismo, nos quais os processadores em questão voltam-se para decidir e realizar as ações que visam o equilíbrio de carga. Em [AM96, FG96, WLR93, ZLP97] são encontrados exemplos de algoritmos distribuídos síncronos.

Quando se trata de um algoritmo *assíncrono*, a sua execução em cada processador é independente da execução nos demais. Cada processador pode disparar a execução do algoritmo no momento em que for necessário, sem que haja sincronismo com execuções em outros processadores. Cabe entretanto ressaltar que um processador, ao iniciar o algoritmo assíncrono, muitas vezes, coleta informações de outros processadores e, ao terminar, divulga sua decisões para que as transferências de carga se realizem entre os processadores envolvidos. Deve ficar claro que esta comunicação entre o processador que executa o algoritmo e os demais não caracteriza a existência de sincronismo entre execuções distribuídas do algoritmo de balanceamento de carga. Algoritmos distribuídos assíncronos são encontrados em [GZ97, KGGK94, WLR93].

Em [WLR93], as classificações síncrona e assíncrona já haviam sido utilizadas com o mesmo significado aqui apresentado.

3.8 Ativação do Algoritmo

A classe das estratégias por migração é também subclassificada quanto aos critérios que definem quando devem ocorrer as ativações do balanceamento de carga.

A periodicidade de ativação, em uma estratégia *periódica*, pode ser definida por tempo corrido, por quantidade de operações realizadas ou a partir de outros índices. Em [AM96], encontra-se um exemplo desta classe de estratégias, no qual a periodicidade de ativação do balanceamento de carga é definida pela quantidade de operações realizadas. Em [FG96, KR97], o balanceamento de carga é ativado após a execução de um conjunto de iterações da aplicação paralela em questão.

A eficiência de uma estratégia periódica está diretamente relacionada à definição adequada do período entre duas ativações do balanceamento de carga. Porém, a identificação deste parâmetro ideal não é um problema simples de ser resolvido. A utilização de um período pequeno entre duas ativações do balanceamento de carga pode acarretar em execuções consecutivas desnecessárias se neste intervalo o equilíbrio não tiver sido ameaçado. O desempenho da aplicação estaria sendo, neste caso, desnecessariamente prejudicado. Por outro lado, o aumento deste período e a conseqüente redução do número de execuções do balanceamento de carga poderia ocasionar a ociosidade de processadores.

Execuções desnecessárias poderiam ocorrer também devido ao dinamismo dos fatores de desequilíbrio, ou seja, estes fatores podem reverter um quadro de desequilíbrio sem que haja a real necessidade de execução do balanceamento. Por exemplo, um processador com uma carga de tarefas elevada, causada por uma sobrecarga externa, pode, sem nenhuma execução do balanceamento de carga, ter a sua carga externa aleatoriamente reduzida e, conseqüentemente, a sua capacidade aumentada, tornando-se posteriormente um processador adequadamente carregado. A execução do balanceamento de carga ativada para corrigir a sobrecarga original de tarefas teria sido desnecessária e seria a principal responsável pela provável falta de tarefas no processador em questão devido à redução do seu número de tarefas, causada pelas transferências do balanceamento e pelo aumento da sua capacidade de processamento.

Em uma estratégia por *evento*, o balanceamento de carga é ativado pela ocorrência de alguma ação ou pela identificação de que alguma condição, previamente definida, tenha sido satisfeita. Estratégias desta classe são encontradas em [GZ97, KGGK94, WLR93, ZLP97]. Exemplos comuns de eventos que disparam o balanceamento de carga são: o fim das tarefas de um determinado processador, utilizado em [GZ97, KGGK94, ZLP97], e a redução da carga de um processador a um nível inferior a um limite mínimo, utilizado em [GZ97, WLR93]. Considera-se que uma estratégia por evento pode estar baseada em trocas periódicas de informações sem que isso a torne uma estratégia periódica, dado que o balanceamento de carga não necessariamente será executado após cada transferência de informações.

3.9 Processadores Alvo

Em um algoritmo de balanceamento de carga por migração, as tranferências definidas visam o equilíbrio de carga de um conjunto de processadores ou visam resolver algum problema de carga sinalizado por um processador, enviando-lhe ou retirando-lhe carga, conforme o problema. De acordo com estas possibilidades, um algoritmo por migração é classificado, respectivamente, como coletivo ou individual.

Em um algoritmo *coletivo*, o objetivo é balancear a carga de um conjunto não unitário de processadores do sistema (eventualmente, todos os processadores do sistema, quando se trata de um algoritmo global). Algoritmos coletivos são encontrados em [AM96, FG96, GZ97, KR97, WLR93, ZLP97].

O algoritmo *individual* procura resolver o problema de um processador que se apresenta com problemas de carga. Por exemplo, um algoritmo individual poderia ser projetado para, sempre que um processador encerrar as suas tarefas, eleger um ou mais processadores, segundo algum critério, para enviar-lhe tarefas. A política de transferência desta classe de algoritmos se particulariza por conter uma **política de localização**, que se refere aos critérios utilizados na definição de quais processadores devem enviar ou receber tarefas do processador com problemas de carga. Em [FG96, GZ97, KGGK94, WLR93] são encontrados exemplos de algoritmos individuais.

Deve ficar clara a diferença entre as classificações global e coletiva. Em uma estratégia global, todos os processadores podem disponibilizar informações para que as transferências referentes ao balanceamento de carga possam ser definidas e todos os processadores podem participar destas transferências. Entretanto, esta estratégia global pode ser individual ou coletiva. Se for individual, as transferências são planejadas apenas para resolver o desequilíbrio de carga de um determinado processador. Se for coletiva, as decisões do balanceamento de carga visam equilibrar a carga de um conjunto de processadores. Uma estratégia local pode ser coletiva se, por exemplo, o balanceamento de carga for executado visando equilibrar a carga de todos os processadores (balanceamento coletivo) de cada grupo (balanceamento local) isoladamente.

3.10 Direção das Transferências

Um algoritmo individual, como definido anteriormente, procura resolver o problema dos processadores que se encontram com carga em excesso ou com menos carga que os demais processadores.

Nas estratégias *receptoras*, as transferências são definidas com o objetivo de enviar carga aos processadores subcarregados. Um processador subcarregado precisa receber tarefas dos processadores mais carregados para que a sua carga se mantenha equilibrada em relação aos demais. Portanto, a política de localização de uma estratégia receptora procura identificar o conjunto de processadores que devem enviar tarefas ao processador subcarregado. Estratégias receptoras podem ser encontradas em [GZ97, KGGK94, WLR93]. Nestes trabalhos, são denominadas *receiver-initiated*.

Nas estratégias *emissoras*, ao contrário, as transferências são definidas com o objetivo de distribuir a carga dos processadores sobrecarregados. Um processador que se apresenta sobrecarregado deve dividir suas tarefas, enviando-as a processadores menos carregados. Desta forma, a política de localização de uma estratégia emissora deve identificar os processadores para os quais o processador sobrecarregado deve enviar tarefas. Exemplos de estratégias emissoras são apresentados em [FG96, WLR93]. Nestes trabalhos, são classificadas como *sender-initiated*.

Esta categorização foi anteriormente utilizada para classificar os algoritmos de balanceamento de carga de sistemas distribuídos [BMD94, LLC96, SKS92]. Neste contexto, uma estratégia de balanceamento de carga *receiver-initiated* é ativada, então, sempre que algum nó se torna subcarregado para que este possa receber novos processos de outros nós. Inversamente, funciona uma estratégia *sender-initiated*, na qual os nós que se tornam sobrecarregados sinalizam a necessidade de exportar alguns processos. Em [BMD94, LLC96, SKS92], considera-se ainda uma forma híbrida, chamada *symmetrically-initiated*, na qual o balanceamento pode ser ativado tanto por um processador subcarregado quanto por um sobrecarregado.

3.11 Política de Localização

Na política de localização de um algoritmo individual, estão definidos os critérios utilizados na identificação dos processadores que devem enviar ou receber tarefas do processador com problemas de carga. Esta identificação pode ser feita tomando-se por base informações sobre a carga dos processadores candidatos, ou seja, considerando-se índices de carga. A partir da utilização ou não deste tipo de informação definem-se duas outras subclasses dos algoritmos individuais: respectivamente, os não cegos e os cegos.

Os algoritmos individuais *não cegos* são aqueles cuja política de localização utiliza índices de carga na identificação dos processadores envolvidos nas transferências. Dependendo do algoritmo, o índice pode ser de carga interna ou externa, ou até mesmo de ambos os tipos. Algoritmos não cegos podem ser encontrados em [FG96, WLR93]. Nestes trabalhos, apenas índices de carga interna foram considerados.

Os algoritmos *cegos* são aqueles que não utilizam índices de carga na sua política de localização. Em vários algoritmos desta classe, os processadores envolvidos nas transferências são escolhidos aleatoriamente ou por rodízio (*round robin*). Exemplos de algoritmos cegos encontram-se em [FG96, GZ97, KGGK94].

4 Utilização da Taxonomia

Nesta seção, serão descritos e classificados alguns algoritmos encontrados na literatura, com o objetivo de ilustrar a utilização da taxonomia proposta.

4.1 Classificação dos Algoritmos Apresentados em [ZLP97]

Em [ZLP97], foi proposto um sistema capaz de selecionar, durante a execução paralela de iterações de um *loop* de uma aplicação, a estratégia mais adequada de balanceamento dinâmico de carga. Foram consideradas quatro variações de uma estratégia de balanceamento dinâmico de carga. Estas variações são combinações dos pares de características: local contra global e centralizado contra distribuído. Em grande parte dos testes realizados, as estratégias selecionadas a partir do modelo de custo definido, apresentaram, realmente, os melhores desempenhos. Os algoritmos distribuídos tiveram, para as aplicações tratadas, um desempenho superior aos centralizados e a eficiência dos algoritmos globais e locais se mostrou dependente da aplicação. Os experimentos mostraram que a escolha da estratégia mais adequada depende de parâmetros como o número de processadores, tamanho dos dados, custo das iterações e custo de comunicação. Foram utilizadas para testes estratégias paralelas de multiplicação de matrizes e de uma aplicação de um *benchmark*. A plataforma computacional adotada foi uma rede de 16 estações de trabalho SUN Sparc LX interconectadas via Ethernet sobre a qual utilizou-se a bilblioteca PVM.

Política de Distribuição

Nesta estratégia de balanceamento de carga, inicialmente, todas as tarefas são homogeneamente distribuídas entre os processadores. Ao longo da execução das tarefas, se necessário, para combater algum desequilíbrio de carga, ocorrem transferências de tarefas dos processadores sobrecarregados para os menos carregados. As transferências são definidas em função das tarefas ainda não realizadas em cada processador e das capacidades apresentadas pelos processadores. Esta capacidade é um reflexo da carga externa à aplicação existente em ambientes como as redes de estações de trabalho. Portanto, trata-se de um algoritmo de balanceamento de carga dinâmico, por migração e integrado.

Políticas de Informação Interna e Externa

O índice de carga interna desta estratégia representa a quantidade de tarefas ainda não executadas em cada processador. O índice de carga externa é definido pelo número de tarefas realizadas por segundo em cada processador desde a última ativação do algoritmo de balanceamento de carga. Estes índices são enviados por cada processador, logo após a ativação do balanceamento de carga, ao processador responsável pela execução do balanceamento (no caso centralizado) ou a todos os processadores (no caso distribuído).

Política de Transferência da Versão 1 - Global e Centralizado

O algoritmo de balanceamento de carga é ativado sempre que algum processador encerra seu conjunto de tarefas. Logo após a ativação, todos os processadores enviam a um processador responsável pelo balanceamento os seus índices de carga (este processador central também executa tarefas ao longo da aplicação). Calcula-se então a nova distribuição de carga necessária ao balanceamento de todo o sistema. As transferências necessárias a este equilíbrio são divulgadas aos processadores envolvidos e estes as realizam. Portanto, trata-se de um algoritmo de balanceamento de carga global, centralizado, por evento e coletivo.

Política de Transferência da Versão 2 - Global e Distribuído

O algoritmo de balanceamento de carga é executado de forma distribuída em todos os processadores. Todos devem enviar os seus índices de carga aos demais processadores logo após o início do algoritmo, que é ativado por algum processador que tenha encerrado as sua tarefas. Em seguida, todos os processadores, ao mesmo tempo, calculam a nova distribuição de carga necessária ao balanceamento de todo o sistema. As transferências necessárias não precisam ser divulgadas, pois todos as identificam. Estas são, então, realizadas pelos respectivos processadores. Trata-se, portanto, de um algoritmo de balanceamento de carga global, distribuído, síncrono por evento e coletivo.

Política de Transferência da Versão 3 - Local e Centralizado

Os processadores são divididos em vários grupos. Cada processador pertence a exatamente um grupo. Não há transferência de carga entre grupos. No início da execução do algoritmo, todos os processadores do grupo do processador que ativou o balanceamento enviam os seus índices ao processador central que executa o algoritmo. Nesta proposta, há um processador central único em vez de um para cada grupo. Este processador central calcula, então, a nova distribuição necessária ao balanceamento de carga do grupo em questão. As transferências necessárias são divulgadas aos processadores envolvidos e estes as realizam. Portanto, trata-se de um algoritmo de balanceamento de carga local, particionado, centralizado, por evento e coletivo.

Política de Transferência da Versão 4 - Local e Distribuído

O mesmo particionamento dos processadores em grupos da versão anterior é considerado. No início da execução do algoritmo, cada processador do grupo daquele que ativou o balanceamento envia os seus índices a todos os outros processadores deste grupo. Em seguida, todos os processadores do grupo em questão calculam a nova distribuição necessária ao balanceamento de carga deste grupo. As transferências necessárias não precisam ser divulgadas, pois os próprios processadores envolvidos as identificam. Estas são, então, realizadas pelos respectivos processadores. Trata-se, portanto, de um algoritmo de balanceamento de carga local, particionado, distribuído, síncrono, por evento e coletivo.

4.2 Classificação do Algoritmo Apresentado em [KR97]

Em [KR97], foram estudadas possíveis otimizações aplicadas a diferentes fases da execução de uma aplicação paralela da classe SPMD quando projetada para ambientes adaptativos e não uniformes, ou seja, nos quais pode haver variação nas condições dos recursos computacionais disponíveis. O balanceamento dinâmico de carga é uma dessas possíveis otimizações consideradas. A aplicação utilizada para teste, uma paralelização de um *loop* para percorrer os nós de um grafo realizando alguns cálculos, foi executada em uma rede de 5 estações de trabalho SUN4 com o sistema P4 de troca de mensagens.

Política de Distribuição

Neste algoritmo de balanceamento de carga, inicialmente, as tarefas são igualmente particionadas entre os processadores. Durante a execução da aplicação, quando necessário, realizam-se transferências de tarefas dos processadores sobrecarregados para os menos carregados. Estas transferências são definidas em função das tarefas ainda não realizadas em cada processador e das capacidades apresentadas por estes. Esta capacidade é um reflexo da carga externa à aplicação existente em ambientes como as redes de estações de trabalho. Portanto, trata-se de um algoritmo de balanceamento de carga dinâmico, por migração e integrado.

Políticas de Informação Interna e Externa

O índice de carga interna deste algoritmo é representado pela quantidade de tarefas ainda não executadas em cada processador. O índice de carga externa é definido pelo tempo médio necessário para a realização de uma tarefa. Estes índices são enviados por cada processador, periodicamente, em cada ativação do algoritmo, ao processador responsável pela execução do balanceamento de carga.

Política de Transferência

O algoritmo de balanceamento de carga é ativado de tempos em tempos, após a execução de um conjunto de tarefas. Logo após a ativação, todos os processadores enviam a um processador responsável pelo balanceamento de carga os seus índices de carga. Calcula-se, então, a nova distribuição de carga necessária ao balanceamento de todo o sistema. As transferências necessárias a este equilíbrio são divulgadas aos processadores envolvidos e estes as realizam. Portanto, trata-se de um algoritmo de balanceamento de carga global, centralizado, periódico e coletivo.

4.3 Classificação dos Algoritmos Apresentados em [FG96]

Em [FG96], foi apresentada uma forma de se representar a distribuição e a transferência de tarefas durante a execução de um algoritmo de balanceamento de carga baseada no modelo iterativo da matriz. Este modelo já havia sido utilizado, em [Cyb89], na análise da convergência do algoritmo *Diffusion*. Em conjunto com este modelo, alguns medidores de desempenho foram definidos para que se possa determinar o custo de execução de uma aplicação paralela e do algoritmo de balanceamento de carga adotado, em um ambiente computacional específico. Foram comparadas as previsões deste modelo de custo com os resultados dos experimentos realizados para o problema de simulação do movimento de particulas, conhecido como N-corpos, utilizando-se três algoritmos de balanceamento de carga: *Random*, *Diffusion* e *Complete Redistribution*. O percentual de erro resultante desta comparação foi menor que 20%. Dos algoritmos considerados, o *Random* apresentou, de uma forma geral, considerando a relação entre custo e desempenho, os melhores resultados. A plataforma computacional adotada foi uma rede de 16 estações de trabalho SUN de capacidades diferentes, variando de SUN 4 até Sparc 20, interconectadas via rede Ethernet. A biblioteca PVM foi utilizada na implementação da aplicação paralela. Apesar de este tipo de plataforma ter sido utilizado, em nenhuma das estratégias de balanceamento de carga adotadas considerou-se a carga externa à aplicação. O desequilíbrio dinâmico considerado se limitou ao causado pela migração de partículas entre as áreas associadas aos diferentes processadores. A quantidade total de partículas permanece constante ao longo da execução da aplicação e deve estar equilibradamente distribuída entre os diversos processadores. O balanceamento de carga é executado apenas para ajustar este equilíbrio em função de constantes migrações destas partículas. Este ajuste, ou seja, o balanceamento de carga, ocorre através de uma nova alocação das regiões do espaço entre os processadores de forma que o conjunto de partículas seja equilibradamente redistribuído. Analisam-se, a seguir, os três algoritmos de balanceamento de carga utilizados neste trabalho.

Política de Distribuição

Nas três estratégias consideradas, inicialmente, todas as tarefas - partículas do sistema - são proporcionalmente distribuídas entre os processadores de acordo com as suas capacidades. Processadores mais rápidos recebem mais tarefas e os de menor capacidade, menos tarefas. Ao longo da execução da aplicação, ou melhor, ao final de cada iteração (cabe lembrar que a simulação dos N-corpos representa uma aplicação iterativa), partículas são transferidas dos processadores sobrecarregados para os menos carregados. As transferências são calculadas em função apenas da carga interna de cada processador. A carga externa à aplicação existente em ambientes como o adotado não foi considerada. Portanto, trata-se de três algoritmos de balanceamento de carga dinâmicos, por migração e isolados.

Política de Informação Interna

O índice de carga interna destas estratégias é representado pela quantidade de partículas dentro da área associada a cada processador. Este índice é enviado por cada processador a todos os outros após o final de cada iteração da aplicação.

Política de Transferência - Random

Na estratégia *random*, após a execução de cada iteração, o balanceamento de carga é ativado. Este é executado em todos os processadores ao mesmo tempo. Os processadores que estiverem sobrecarregados, devem escolher, aleatoriamente, entre seus vizinhos, o processador que irá receber parte das suas partículas. Esta vizinhança é definida pela aplicação (e não pelo topologia da rede) de acordo com as regiões do espaço alocadas aos processadores. Trata-se, portanto, de um algoritmo de balanceamento de carga local, por vizinhança, distribuído, síncrono, individual e periódico.

Política de Localização - Random

O algoritmo *random* é individual e, portanto, possui uma política de localização. Conforme, descrito, quando um processador encontra-se sobrecarregado, escolhe, aleatoriamente, o processador para o qual enviará parte de suas tarefas. Portanto, esta política é caracterizada pela aleatoriedade da localização do processador destino. Trata-se, portanto, de um algoritmo de balanceamento de carga emissor e cego.

Política de Transferência - Diffusion

Na estratégia *diffusion*, o balanceamento de carga também é ativado em todos os processadores após a execução de cada iteração. Cada processador calcula a média de partículas dos seus vizinhos e, se a sua carga interna for alta em relação a essa média, envia partículas aos menos carregados. Esta vizinhança é a mesma definida pelas regiões do espaço alocadas aos processadores. Trata-se, portanto, de um algoritmo de balanceamento de carga local, por vizinhança, distribuído, síncrono, individual e periódico.

Política de Localização - Diffusion

O algoritmo *diffusion*, por ser individual, também possui uma política de localização. Esta estabelece que cada processador sobrecarregado deve enviar tarefas aos processadores vizinhos que possuam menos partículas que a média de partículas calculada. Utiliza-se, desta forma, informações sobre a carga dos vizinhos na localização dos processadores aos quais devem ser enviadas tarefas. Trata-se, portanto, de um algoritmo de balanceamento de carga emissor e não cego.

Política de Transferência - Complete Redistribution

Na estratégia *complete redistribution*, após a execução de cada iteração, o balanceamento de carga é executado por um processador central. Sempre que este identifica que algum processador encontra-se sobrecarregado, há uma redistribuição de todas as tarefas entre os processadores. O processador central deve definir a nova região do espaço a ser atribuída a cada processador. Trata-se, portanto, de um algoritmo de balanceamento de carga global, centralizado, coletivo e periódico.

4.4 Classificação do Algoritmo Apresentado em [LPR97]

Em [LPR97], encontra-se a utilização de uma estratégia de balanceamento de carga, chamada *Dose-Driven*, na execução paralela de consultas recursivas a bancos de dados (cálculo paralelo do fecho transitivo de relações binárias). Cada tarefa representa o cálculo dos sucessores de um conjunto de nós. Procura-se, neste trabalho, anular os efeitos de fatores dinâmicos de desequilíbrio como a variação e heterogeneidade da carga externa à aplicação e o não conhecimento prévio da demanda computacional de cada tarefa. Os testes computacionais foram realizados em um computador IBM 9076 SP/2 com 16 nós. A biblioteca MPI foi utilizada na implementação da aplicação paralela. Os resultados computacionais indicam que a eficiência do tipo de estratégia utilizada está relacionada à definição adequada da granularidade das tarefas.

Política de Distribuição

Nesta estratégia, as tarefas são distribuídas ao longo da aplicação. Um processador, eleito o coordenador, controla a distribuição das tarefas e os demais processadores as executam.

Inicialmente, o coordenador envia uma tarefa a cada processador. Enquanto houver tarefas, cada processador recebe uma nova após a conclusão da tarefa corrente. A carga externa à aplicação existente em ambientes como o adotado não foi considerada pela estratégia. Portanto, trata-se de um algoritmo de balanceamento de carga dinâmico, sob demanda e isolado.

4.5 Classificação de um Algoritmo Apresentado em [WLR93]

Em [WLR93], foram comparadas cinco estratégias de balanceamento dinâmico de carga: *Sender Initiated Diffusion (SID)*, *Receiver Initiated Diffusion (RID)*, *Hierarchical Balancing Method (HBM)*, *Gradient Model (GM)* e *Dimension Exchange Method (DEM)*. Algumas dessas estratégias já haviam sido utilizadas, em trabalhos anteriores, com o objetivo de balancear a carga dos diferentes nós de um sistema distribuído. Neste trabalho, aparecem visando balancear a carga de uma aplicação paralela. Duas aplicações foram utilizadas para teste: uma estratégia artificial composta por um conjunto de tarefas com demanda computacional aleatória e uma implementação da técnica *branch-and-bound* para o problema de escalonamento de processos. De acordo com os testes realizados em um computador hipercubo Intel iPSC/2 com 32 processadores, concluiu-se que a estratégia RID pode ser mais adequada a um conjunto maior de ambientes e aplicações. Segue uma caracterização desta estratégia a partir da sua classificação.

A estratégia RID é **dinâmica** e **por migração**, portanto, inicialmente, as tarefas da aplicação são todas distribuídas e posteriormente, caso seja necessário, são transferidas entre os processadores para um ajuste de carga. Além disso, é **isolada**, ou seja, as transferências são calculadas em função apenas da carga interna de cada processador. A carga externa à aplicação não foi considerada. O índice de carga interna adotado representa a quantidade de tarefas ainda não executadas em cada processador. Esta informação é divulgada periodicamente.

A estratégia é **local** e **por vizinhança**. Desta forma, as transferências são feitas apenas entre grupos de processadores definidos pela vizinhança entre estes. Neste caso, a vizinhança foi definida pela topologia do sistema de interconexão - um hipercubo.

O balanceamento de carga é ativado **por evento**. Neste caso, por qualquer processador que se torne subcarregado.

A execução da estratégia é **distribuída** e **assíncrona**. O balanceamento de carga é executado pelo processador que o ativou. Cada processador pode ativar a estratégia no momento em que for necessário sem que haja sincronismo com execuções desta em outros processadores.

A estratégia RID é **individual**, **receptora** e **não cega**. Portanto, a estratégia visa resolver o problema de carga de um processador - aquele que se torna subcarregado e ativa o balanceamento. Um subconjunto de processadores com carga em excesso deve ser identificado para enviar tarefas ao subcarregado. A identificação destes processadores e da quantidade de tarefas a serem transferidas de cada um é realizada em função das cargas dos processadores vizinhos.

4.6 Classificação do Algoritmo Apresentado em [LT92]

Há algoritmos que combinam características de classes complementares da taxonomia. Em [LT92], por exemplo, encontra-se uma estratégia de balanceamento dinâmico de carga que utiliza, em momentos diferentes, políticas de ações de classes complementares. Esta estratégia foi utilizada no cálculo paralelo da junção relacional, uma aplicação da área de banco de dados. Inicialmente, é executada com uma política de distribuição sob demanda, ou seja, blocos de tarefas são distribuídos ao longo da execução da aplicação por um processador coordenador aos outros. Enquanto há tarefas a serem realizadas, cada processador, ao terminar as tarefas recebidas, requisita outro bloco ao coordenador. Após a distribuição dos últimos blocos, a estratégia passa a executar uma política por migração, ou seja, a executar transferências de tarefas dos mais carregados aos menos carregados. Portanto, este é um exemplo de uma estratégia que pode ser classificada, de forma híbrida, como sob demanda e por migração.

5 Considerações Finais

Propôs-se, neste trabalho, uma taxonomia de algoritmos de balanceamento de carga projetados para aplicações SPMD.

Esta taxonomia viabiliza uma classificação e uma conseqüente organização em categorias do crescente conjunto de algoritmos de balanceamento de carga projetados para aplicações SPMD em arquiteturas MIMD com memória distribuída. Diferentes algoritmos de balanceamento de carga utilizados em aplicações de diversas áreas foram categorizados a partir da taxonomia.

Uma vez conhecidas as classes que compõem a taxonomia, um algoritmo de balanceamento de carga pode ser naturalmente descrito e facilmente entendido utilizando-se a terminologia proposta. Para ilustrar esta propriedade da taxonomia, uma estratégia foi caracterizada, na Subseção 4.5, a partir da sua classificação.

Estas características e propriedades da taxonomia proposta facilitam a comparação mais criteriosa entre estratégias, auxiliando projetistas de aplicações paralelas na tarefa de identificar estratégias de balanceamento de carga adequadas à aplicação em desenvolvimento.

Referências Bibliográficas

[AM96] J. N. C. Árabe e C. D. Murta, "Auto-balanceamento de carga em programas paralelos", *Anais do VIII Simpósio Brasileiro de Arquitetura de Computadores e Processamento de Alto Desempenho*, 1996, 161–171.

[BMD94] K. Benmohammed-Mahieddine e P. M. Dew, "A periodic symmetrically-initiated load balancing algorithm for distributed systems", *Operating Systems Review* 28 (1994), 66–77.

[BS96] L. Baker e B.J. Smith, *Parallel Programming*, McGraw-Hill, 1996.

[BT97] D. P. Bertsekas e J. N. Tsitsiklis, *Parallel and Distributed Computation: Numerical Methods*, Athena Scientific, 1997.

[CCH93] F. Cacace, S. Ceri e M. Houtsma, "A survey of parallel execution strategies for transitive closure and logic programs", *Distributed and Parallel Databases* 1 (1993), 337–382.

[CK88] T. L. Casavant e J. G. Kuhl, "A taxonomy of scheduling in general-purpose distributed computing systems", *IEEE Transactions on Software Engineering* 14 (1998), 141–154.

[Cyb89] G. Cybenko, "Dynamic load balancing for distributed memory multiprocessors", *Journal of Parallel and Distributed Computing* 7 (1989), 279–301.

[dBS97] R. de B. Seixas, *Visualização Volumétrica com Ray-Casting num Ambiente Distribuído*, Tese de doutorado, Departamento de Informática, Pontifícia Universidade Católica, Rio de Janeiro, Brasil, 1997.

[Duc96] R. Ducan, "MIMD architectures: Shared and distributed memory desings", *Parallel and Distributed Computing Handbook* (A. Y. Zomaya, editor), Series on Computer Engineering, 680–698, McGraw-Hill, 1996.

[ELZ86] D. L. Eager, E. D. Lazowska e J. Zahorjan, "Adaptive load sharing in homogeneous distributed systems", *IEEE Transactions on Software Engineering* SE-12 (1986), 662–675.

[Eri96] H. Erika, "New results of the Salzburg NTN-method for the Radon transform", *Proceedings of the Third International ACPC Conference with Special Emphasis on Parallel Databases and Parallel I/O*, Klagenfurt, Springer-Verlag, Lecture Notes in Computer Science 1127 (1996), 62–77.

[FG96] M. A. Franklin e V. Govindan, "A general matrix iterative model for dynamic load balancing", *Parallel Computing* 22 (1996), 969–989.

[Fos94] I. Foster, *Designing and Building Parallel Programs: Concepts and Tools for Parallel Software Engineering*, Addison-Wesley, 1994.

[FTI90] M. Furuichi, K. Taki e N. Ichiyoshi, "A multi-level load balancing scheme for or-parallel exhaustive search programs on the multi-psi", *Proceedings of the Second ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, 1990, 50–59.

[GZ97] N. Guil e E. L. Zapata, "Fast Hough transform on multiprocessors: A branch and bound approach", *Journal of Parallel and Distributed Computing* 45 (1997), 82–89.

[Han95] P. B. Hansen, *Studies in Computational Science: Parallel Programming Paradigms*, Prentice Hall, 1995.

[KGGK94] V. Kumar, A. Grama, A. Gupta e G. Karypis, *Introduction to Parallel Computing: Desing and Analysis of Parallel Algorithms*, Benjamin/Cummings, 1994.

[KR97] M. Kaddoura e S. Ranka, "Runtime support for parallelization of data-parallel applications on adaptive and nonuniform computational environment", *Journal of Parallel and Distributed Computing* 43 (1997), 163–168.

[LLC96] G. Lee, H. Lee e J. Cho, "A prediction-based adaptive location policy for distributed load balancing", *Journal of Systems Architecture* 42 (1996), 1–18.

[LOT94] H. Lu, B. Ooi e K. Tan, *Query Processing in Parallel Relational Database Systems*, capítulo 4, 147–225, IEEE Computer Society Press, 1994.

[LPR97] S. Lifschitz, A. Plastino e C. C. Ribeiro, "Exploring load balancing in parallel processing of recursive queries", *Proceedings of the Third International Euro-Par Parallel Processing Conference*, Passau, Springer-Verlag, Lecture Notes in Computer Science 1300 (1997), 1125–1129.

[LRN97] T. Lee, C. S. Raghavendra e J. B. Nicholas, "Parallel implementation of a ray tracing algorithm for distributed memory parallel computers", *Concurrency: Practice and Experience* 9 (1997), 947–965.

[LT92] H. Lu e K. Tan, “Dynamic and load-balanced task-oriented database query processing in parallel systems”, *Proceedings of the Third International Conference on Extending Data Base Technology*, Springer-Verlag, 1992, 357–372.

[Mat96] T. G. Mattson, “Scientific computation”, *Parallel and Distributed Computing Handbook* (A. Y. Zomaya, editor), Series on Computer Engineering, 981–1002, McGraw-Hill, 1996.

[PPMR95] P. M. Pardalos, L. Pitsoulis, T. Mavridou e M. G. C. Resende, “Parallel search for combinatorial optimization: Genetic algorithms, simulated anneling, tabu search and grasp”, *Proceedings of the Workshop on Parallel Algorithms for Irregularly Structured Problems*, Lyon, 1995, 317–331.

[Qui94] M. J. Quinn, *Parallel Computing: Theory and Practice*, Series in Computer Science, McGraw-Hill, 1994.

[SHK95] B. A. Shirazi, A. R. Hurson e K. M. Kavi, *Scheduling and Load Balancing in Parallel and Distributed Systems*, IEEE Computer Society Press, 1995.

[SKS92] N. G. Shivaratri, P. Krueger e M. Singhal, “Load distributing for locally distributed systems”, *Computer* 25 (1992), 33–44.

[WLR93] M. A. Willebeek-LeMair e A. P. Reeves, “Strategies for dynamic load balancing on highly parallel computers”, *IEEE Transactions on Parallel and Distributed Systems* 4 (1993), 979–993.

[ZLP97] M. J. Zaki, W. Li e S. Parthasarathy, “Customized dynamic load balancing for a network of workstations”, *Journal of Parallel and Distributed Computing* 43 (1997), 156–162.

[Zom96] A. Y. Zomaya, “Parallel and distributed computing: The scene, the props, the players”, *Parallel and Distributed Computing Handbook* (A. Y. Zomaya, editor), Series on Computer Engineering, 5–23, McGraw-Hill, 1996.