UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

RICARDO VARGAS DORNELES

Particionamento de Domínio e Balanceamento de Carga no Modelo HIDRA

Tese apresentada como requisito parcial
para a obtenção do grau de
Doutor em Ciência da Computação

Prof. Dr. Tiarajú Asmuz Diverio
Orientador

Prof. Dr. Philippe Olivier Alexandre Navaux
Co-orientador

Porto Alegre, junho de 2003

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Dorneles, Ricardo Vargas

Particionamento de Domínio e Balanceamento de Carga no Modelo HIDRA / Ricardo Vargas Dorneles. – Porto Alegre: Programa de Pós-Graduação em Computação, 2003.

136 f.: il.

Tese (doutorado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2003. Orientador: Tiarajú Asmuz Diverio; Co-orientador: Philippe Olivier Alexandre Navaux.

1. Processamento Paralelo. 2. Paralelização de Métodos Numéricos. 3. Balanceamento Dinâmico de Carga. 4. Particionamento de Domínio. I. Diverio, Tiarajú Asmuz. II. Navaux, Philippe Olivier Alexandre. III. Título.

À Maristela e à Catherine.

AGRADECIMENTOS

Para que esse trabalho chegasse ao fim foi necessário o apoio de diversas pessoas. Quero deixar registrados meus agradecimentos:

Ao Programa de Pós Graduação em Ciência da Computação do Instituto de Informática da UFRGS, por todos os recursos disponibilizados.

Aos professores do PPGC pelo que aprendi com eles.

Ao meu orientador Prof. Dr. Tiarajú Asmuz Diverio pela amizade, por ter me aceito como seu orientando e pela orientação durante esse período.

Ao meu co-orientador, Prof. Dr. Philippe Navaux, pela amizade, por ter aceito co-orientar este trabalho e pela disposição com que sempre me recebeu quando precisei nesse período.

Ao meu amigo Rogerio Rizzi, matemático paciencioso e garimpador incansável de artigos, pelo que aprendi com ele sobre discretização, EDPs e pela paciência com meus horários nem sempre muito ortodoxos.

Ao Prof. Dr. Renato Simões Silva, pelo apoio e valiosas sugestões por ocasião da defesa da proposta de tese e por disponibilizar o acesso ao *cluster* do LNCC, essencial para os primeiros experimentos sobre escalabilidade do modelo.

Aos amigos que tornaram esse período, além de muito produtivo e enriquecedor, também muito agradável. Ana Paula, Cassal, Marilton, Patty, Wives, Marcia Moraes, Zeferino, Pilla, Denise, Monica, Bohrer, Barreto, Marcia Pasin, Ju, Delcino e Martinotto.

Ao pessoal dos *clusters*, representados aqui, inicialmente, pelo Bohrer e o Pilla, e hoje pelo Sanger, Diego e Clarissa, pela boa vontade com que sempre tiraram minhas dúvidas e atenderam minhas solicitações.

À Universidade de Caxias pelas horas concedidas para o desenvolvimento desse trabalho. Ao Marcos Casa, Cristian Koliver, Heitor Strogulski e todos os amigos de lá pelo bom ambiente de trabalho.

Ao pessoal da Biblioteca, Ida, Beatriz, Henrique, Adriana e Greice pela simpatia e presteza com que me receberam nesses anos. Ao pessoal da segurança, representados pelo seu Astrogildo e pelo Luciano, pela importância do trabalho que realizam.

À minha mãe, Olga Vargas Dorneles, e meu pai, Antonio José Beck Dorneles, por tudo que fizeram por mim e por terem, desde cedo, me passado o amor pelo estudo. A meus irmãos, Roberto e Beatriz, pelo ótimo ambiente familiar sempre proporcionado.

E, para finalizar, à Maristela e à Cathy, pelo amor ilimitado que tenho por ambas e pelas horas que lhes tirei para que esse trabalho pudesse ser concluído.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ADI	*Alternating Directions Implicit*
CPU	*Central Processing Unit*
GC	Gradiente Conjugado
GMCPAD	Grupo de Matemática da Computação e Processamento de Alto Desempenho
GMRES	Método do Resíduo Mínimo Generalizado
GPPD	Grupo de Processamento Paralelo e Distribuído
EDP	Equação Diferencial Parcial
ESW	Equações de Shallow Water
ETM	Equação de Transporte de Massa
ORB	*Orthogonal Recursive Bisection*
RCB	*Recursive Coordinate Bisection*
RDPMC	*Read Performance Monitoring Counters*
SDP	Simétrica Definida Positiva
SPMD	*Single Program Multiple Data*
SRCB	*Straight Recursive Coordinate Bisection*
SSTRIP	*Straight Stripwise Partitioning*
STRIP	*Stripwise Partitioning*

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A paralelização de aplicações envolvendo a solução de problemas definidos sob o escopo da Dinâmica dos Fluidos Computacional normalmente é obtida via paralelismo de dados, onde o domínio da aplicação é dividido entre os diversos processadores. Essa divisão do domínio de forma balanceada entre os processadores, bem como a manutenção do balanceamento durante a execução é um problema complexo e diversas heurísticas têm sido desenvolvidas. Aplicações onde a simulação é dividida em diversas fases sobre partes diferentes do domínio acrescentam uma dificuldade maior ao particionamento, ao se buscar a distribuição equilibrada das cargas em todas as fases. Este trabalho descreve a implementação de mecanismos de particionamento e balanceamento de carga em problemas multi-fase sobre *clusters* de PCs. Inicialmente é apresentada a aplicação desenvolvida, um modelo de circulação e transporte de substâncias sobre corpos hídricos 2D e 3D, que pode ser utilizado para modelar qualquer corpo hídrico a partir da descrição de sua geometria, batimetria e condições de contorno. Todo o desenvolvimento e testes do modelo foi feito utilizando como caso de estudo o domínio do Lago Guaíba, em Porto Alegre. Após, são descritas as principais heurísticas de particionamento de domínio de aplicações multi-fase em *clusters*, bem como mecanismos para balanceamento de carga para este tipo de aplicação. Ao final, é apresentada a solução proposta e desenvolvida, bem como os resultados obtidos com a mesma.

Palavras-chave: Processamento Paralelo, Paralelização de Métodos Numéricos, Balanceamento Dinâmico de Carga, Particionamento de Domínio.

Domain Partitioning and Dynamic Load Balancing

ABSTRACT

The parallelization of applications in the area of Computational Fluid Mechanics is usually obtained by data parallelism, where the application domain is divided between several processors. The problem of getting a balanced division of the domain between the processors and keeping the load balance during execution is complex and several heuristics have been developed. Applications where the simulation is divided in several phases over different parts of the domain aggregate a greater complexity to the partitioning, when a balanced distribution along all the phases is searched. This work describes the implementation of partitioning and load balancing mechanisms in multi-phase problems. Firstly, the application being developed, a model of hydrodinamics and mass transport over 2D and 3D hydric bodies, is presented. All the development and tests of the model were made using the domain of Guaiba Lake, in Porto Alegre. After that, the main heuristics of domain partitioning of multi-phase applications on clusters are described, as well as the mechanisms for load balancing of this type of applications. Finally, the solution developed is presented, as well as the performance results obtained.

Keywords: Dynamic Load Balancing, Domain Decomposition, Parallel Processing.

1 INTRODUÇÃO

Na modelagem de fenômenos físicos envolvidos em mecânica de fluidos, os mesmos são descritos através de sistemas de equações diferenciais parciais (EDPs) acopladas que geralmente não possuem solução analítica. Assim, a solução numérica é a alternativa mais adequada de resolver os problemas. Para isso, o domínio contínuo é discretizado, gerando uma grade ou malha de pontos, e as equações contínuas também são discretizadas gerando sistemas de equações que são resolvidos nessas malhas. Além da discretização espacial, o tempo também é discretizado sendo para isso dividido em passos e, para cada passo de tempo, as equações são resolvidas.

A forma de paralelização mais utilizada em modelos hidrodinâmicos é a divisão do domínio entre os diversos processadores. A partir dessa distribuição, algumas abordagens são possíveis. Em uma delas, cada processador monta, a partir de seu subdomínio, um sistema de equações e os diversos sistemas são resolvidos pelos processadores onde foram gerados, trocam-se dados nas fronteiras dos subdomínios e o processo se repete até que as soluções dos sistemas locais convergem para a solução global. Outra abordagem possível é aquela na qual é montado um único sistema para todo o domínio, cada processador montando uma parte do sistema, e o mesmo sendo resolvido em paralelo pelos diversos processadores. Além disso, podem ser utilizadas abordagens híbridas, como os métodos de Krylov-Schwarz (RIZZI, 2002), onde um *solver* paralelo é acelerado por um pré-condicionador. Qualquer que seja o método utilizado, para manter a consistência e a continuidade na fronteira entre os subdomínios, os processadores precisam trocar dados entre si a cada passo de tempo, o que insere pontos de sincronização entre os mesmos.

Um aspecto importante nessas formas de paralelização é o modo como o domínio é particionado entre os processadores. Uma distribuição desigual do domínio entre os processadores pode fazer com que alguns terminem antes sua tarefa, permanecendo ociosos até que os outros processadores cheguem nos pontos de sincronização e os primeiros recebam os dados necessários para executar uma nova carga de tarefas, o que resulta em redução da eficiência. O problema de balanceamento de carga em uma aplicação paralela consiste em atribuir cargas tais aos diferentes processadores de forma que levem tempos aproximadamente iguais para executar sua parte da tarefa. Em aplicações onde é necessária a troca de dados de fronteira entre os subdomínios, como em modelos hidrodinâmicos, é importante, além de manter o equilíbrio de carga, que o particionamento seja feito de forma a reduzir as fronteiras entre os processadores, minimizando, conseqüentemente, a comunicação entre os mesmos. Esse problema é um problema NP-difícil [1] (GAREY; JOHNSON, ????)

[1] Um problema é NP-Completo se é NP, ou seja, uma solução para o mesmo pode ser verificada

e tem sido exaustivamente pesquisado na literatura. O capítulo 2 desse trabalho apresenta os principais conceitos e algoritmos da área.

Na simulação de fenômenos físicos pode ocorrer que um particionamento de domínio que, inicialmente, apresenta um bom balanceamento de carga, mantenha essa característica durante todo o tempo de execução do modelo. Isto ocorre por exemplo, em modelos de hidrodinâmica em que a quantidade de cálculos por célula bem como o número de células por subdomínio são constantes durante todo o tempo de execução. Outra situação ocorre se forem modelados fenômenos físicos que acarretem diferentes quantidades de processamento em diferentes pontos do subdomínio. Um exemplo de fenômeno físico nessa situação é a radiação solar em modelos atmosféricos globais (FOSTER, 1994), a qual é calculada somente nas áreas onde ocorre incidência de luz solar a cada hora do dia. Outra situação, essa abordada nesse trabalho, é a modelagem de transporte de poluentes, onde ocorre emissão de poluentes somente em uma parte do domínio global, devendo essas regiões serem modeladas com um refinamento maior que o resto do domínio. Esse refinamento é efetuado durante a execução do modelo, acarretando alterações na configuração do mesmo, o que pode levar a eventuais desbalanceamentos e a correção dos mesmos é feita através de mecanismos de balanceamento dinâmico de carga.

Uma alternativa cada vez mais comum para obtenção de desempenho a um custo razoável, ao se avaliar as alternativas de *hardware* existentes, é o uso de *clusters* de computadores. O desenvolvimento e barateamento da tecnologia de redes locais de alta velocidade e a evolução na capacidade de processamento dos computadores pessoais tornou possível a montagem de máquinas paralelas de custo acessível com um desempenho da ordem de Gigaflops. Hoje diversas instituições de pesquisa possuem *clusters* com o número de nodos variando desde 2 até centenas de nodos. No *site* http://www.top500.org, de atualização semestral, que lista as 500 instalações de maior capacidade de processamento do mundo, a participação de *clusters* tem aumentado a cada edição. O *site* http://cluster.top500.org mantem uma base de dados com dados sobre instalações de *clusters* em todo o mundo.

Quando todos os nodos de um *cluster* possuem as mesmas características e a rede de comunicação interligando os nodos é a mesma, o *cluster* é dito homogêneo. Se o *cluster* possui nodos com diferentes características ou diferentes redes de comunicação entre grupos de nodos, o *cluster* é dito heterogêneo. A característica de um *cluster* ser ou não homogêneo tem implicações importantes no balanceamento de carga de aplicações e isto será discutido ao longo deste trabalho.

Os nodos de um *cluster* podem ser monoprocessados ou multiprocessados. Quando são utilizados nodos multiprocessados, a comunicação entre os processadores do mesmo nodo geralmente é feita via memória compartilhada, enquanto a comunicação entre os diferentes nodos é feita por troca de mensagens. *Clusters* com nodos multiprocessados são chamados de *clusters* híbridos, por utilizarem ao mesmo tempo memória compartilhada e memória distribuída.

O *cluster* utilizado na fase final do desenvolvimento deste trabalho foi o cluster Labtec, do Instituto de Informática da UFRGS, composto por 20 nodos duais, e

em tempo polinomial (Non-deterministic Polinomial) e se todos os problemas em NP podem ser reduzidos a ele por uma transformação polinomial. Um problema NP-Difícil é um problema para o qual todos os problemas em NP podem ser reduzidos, sem ser necessariamente NP (ou seja, uma solução para o mesmo não é necessariamente verificável em tempo polinomial). Assim sendo, podem ser considerados "mais difíceis"que os NP-Completos. Normalmente problemas de otimização recaem nessa categoria.

cuja descrição encontra-se na seção 4.1.7. Além dele, foi utilizado na fase inicial do desenvolvimento um *cluster* de 11 nodos sobre o qual foi desenvolvido o DECK, uma biblioteca de troca de mensagens (BARRETO et al., 2000), e que atualmente está desativado.

O GMCPAD - Grupo de Matemática Computacional e Processamento de Alto Desempenho e o GPPD - Grupo de Processamento Paralelo e Distribuído da UFRGS vêm trabalhando no desenvolvimento de modelos paralelos de hidrodinâmica e transporte (DORNELES et al., 2000)(RIZZI et al., 2000), e paralelização de métodos numéricos (CANAL, 2001)(PICININ, 2001) sobre este *cluster*. Até o momento foram implementados diversos modelos hidrológicos e de transporte de massa, com diferentes discretizações e utilizando diversos métodos numéricos para solução, os quais são apresentados resumidamente aqui. Descrições mais aprofundadas dos mesmos podem ser encontradas em (DORNELES et al., 2000) (DORNELES et al., 2000) (RIZZI et al., 2000) (RIZZI et al., 2001) e (RIZZI, 2002).

Este trabalho está organizado da seguinte maneira. No capítulo 2 são apresentados os principais conceitos envolvidos em particionamento de domínio com uma visão geral de algoritmos e bibliotecas de particionamento de grafos, com uma ênfase maior naqueles que foram utilizados nesse trabalho. No capítulo 3 é apresentado o problema de balanceamento dinâmico de carga de modo geral bem como o problema específico de modelos multi-fase. São apresentadas algumas abordagens possíveis, entre elas a difusão, utilizada nesse trabalho. No capítulo 4 é apresentado o modelo de hidrodinâmica e transporte desenvolvido nesse trabalho e em (RIZZI, 2002), sobre o qual foram implementadas as soluções de particionamento e balanceamento apresentadas nesse trabalho. São apresentadas as equações de Shallow Water e a Equação de Transporte de Substâncias 3D, nas quais se baseia o modelo de hidrodinâmica e transporte desenvolvido. São apresentadas também as diferentes discretizações utilizadas ao longo da evolução do modelo e os métodos de resolução utilizados para resolver os sistemas de equações gerados. O conjunto das equações, juntamente com os métodos de discretização escolhidos, os métodos numéricos utilizados para resolvê-los e os mecanismos de particionamento e balanceamento incorporados constituem o modelo paralelo de hidrodinâmica e transporte paralelo citado acima que foi resultado desse trabalho e da tese de doutorado de Rogerio Rizzi (RIZZI, 2002), modelo esse que será chamado, ao longo desse trabalho, de HIDRA 2D e HIDRA 3D, ou simplesmente HIDRA, no que for comum a ambos. No capítulo 5 são apresentadas as diversas soluções adotadas para o particionamento estático da hidrodinâmica do HIDRA e no capítulo 6 são apresentadas as soluções implementadas para o balanceamento dinâmico do transporte. No capítulo 7 são apresentados e discutidos os resultados obtidos com cada uma das soluções implementadas. Por fim, no capítulo 8 é apresentado um panorama geral do que foi desenvolvido ao longo desse trabalho bem como algumas conclusões resultantes do mesmo e algumas sugestões de continuidade a esse trabalho.

2 PARTICIONAMENTO DE DOMÍNIO

Neste capítulo serão abordados o problema de balanceamento de carga estático e o problema de particionamento de domínio. Em problemas de simulação como o tratado neste trabalho, o paralelismo é obtido através do particionamento do problema entre os diversos processadores. Diversos algoritmos com essa finalidade serão estudados nesse capítulo. Para que a simulação seja feita com um máximo de eficiência, é necessário que a carga de trabalho de cada processador, proporcional à parte do domínio por ele tratada, seja proporcional a sua capacidade de processamento, ou seja, que a carga de processamento do problema seja bem balanceada entre os processadores.

Serão apresentados aqui os principais algoritmos de particionamento de domínio, bem como os principais conceitos relativos a balanceamento de carga estático. No capítulo 5 será apresentada a forma como está implementado o particionamento de domínio no HIDRA. No capítulo final deste trabalho serão descritos e avaliados os mecanismos de balanceamento dinâmico implementados no HIDRA.

Balanceamento de carga em um problema de EDP em uma máquina paralela pode ser visto como um problema em duas etapas (HOUSTIS, 1995). Em primeiro lugar, a computação é decomposta em tarefas balanceadas requerendo um mínimo de sincronização e de comunicação entre si e, em segundo lugar, as tarefas devem ser atribuídas aos processadores.

Existem duas abordagens para a divisão do problema entre os processadores. Na primeira delas, a malha ou grade resultante da discretização é dividida entre os processadores, e cada processador resolve localmente o sistema resultante de seu subdomínio. Em uma outra abordagem, a malha ou grade é igualmente distribuída entre os processadores, é gerado um único sistema para todo o domínio, também distribuído entre os processadores, e o sistema é resolvido de forma distribuída por todos os processadores, através de um resolvedor (*solver*) paralelo.

No trabalho que vem sendo desenvolvido pelo GMCPAD e GPPD foram implementados métodos utilizando ambas as abordagens. Devido à predominância dessa abordagem na literatura, as versões do HIDRA com suporte para subdomínios irregulares basearam-se na abordagem de decomposição de domínio. Entretanto, versões do HIDRA baseadas em particionamento de dados com a implementação de *solvers* paralelos estão sendo desenvolvidas no grupo.

O modelo de programação utilizado para tais computações é o modelo SPMD (*single program multiple data*) onde todos os processadores executam o mesmo código e o paralelismo é obtido pelo particionamento das estruturas (grade/malha) do problema e pela alocação aos diferentes processadores. Durante cada iteração os processadores executam: (i) uma troca de dados da fronteira com os processadores

que contêm subdomínios adjacentes para atender às condições de continuidade para a solução das EDPs; (ii) uma execução de computação local; (iii) uma avaliação do critério de parada e aceleração da convergência (sincronização global).

O bom desempenho destas computações em máquinas de memória distribuída depende da minimização do tempo de comunicação que depende, por sua vez, da implementação eficiente das operações de comunicação em hardware e software.

As estruturas geométricas utilizadas para representar o domínio na discretização de problemas de EDPs são grades estruturadas quando é utilizado o método de diferenças finitas, ou malhas (grades estruturadas ou não), quando utilizado o método dos elementos finitos. Em ambos os casos, os métodos de particionamento e mapeamento são praticamente os mesmos (HOUSTIS, 1995).

Para efetuar um balanceamento de carga apropriado deve-se levar em conta o custo computacional das tarefas, a dependência entre as mesmas, o custo de comunicação e a forma de interconexão entre os processadores. Além disso, deve ser avaliado o custo da redistribuição da carga em comparação com o ganho obtido para verificar quando deve ser feita a redistribuição.

Balanceamento de Carga pode ser estático, semi-estático ou dinâmico (DEMMEL, 1999a). O primeiro caso ocorre quando toda a informação para o balanceamento está disponível antes do processamento. Esta situação ocorre, por exemplo, em modelos hidrodinâmicos sem refinamento dinâmico ou malhas adaptativas, onde é possível calcular a carga inicial em cada processador e essa carga se mantém fixa durante todo o tempo de processamento.

Balanceamento semi-estático é utilizado em situações onde o domínio é alterado lentamente ao longo do tempo. Neste caso, pode-se utilizar um algoritmo estático no início da aplicação e permitir que a mesma possua algum desbalanceamento após um certo número de passos, quando então é aplicado novamente o algoritmo estático.

Balanceamento dinâmico é utilizado em situações onde a carga dos processadores varia durante a execução tornando necessário a redistribuição de carga para manter o equilíbrio. Alguns autores não diferenciam balanceamento semi-estático de balanceamento dinâmico.

Fox (FOX; WILLIAMS; MESSINA, 1994) considera ainda uma quarta categoria de balanceamento, a qual chama de balanceamento por inspeção, quando o mesmo é trivial e pode ser calculado já na programação da aplicação.

Algoritmos de balanceamento dinâmico de carga podem ser centralizados, totalmente distribuídos ou semi-distribuídos ((ZNATI; MELHEM; PRUHS, 1991), citado por (QUINN, 1994)).

Algoritmos centralizados normalmente são executados por um único processador ou nodo e tomam decisões globais sobre a realocação de trabalho aos processadores. Podem efetuar um bom balanceamento de carga, mas sua escalabilidade é baixa, uma vez que a quantidade de informação necessária para o balanceamento normalmente cresce linearmente com o número de processadores.

Nos algoritmos totalmente distribuídos cada processador constrói sua própria visão da distribuição da carga. Processadores trocam informações com os processadores vizinhos e usam essa informação para fazer alterações locais na alocação da carga. Como as comunicações são basicamente com os processadores vizinhos, esses algoritmos têm a vantagem de gerar um *overhead* de comunicação menor. Entretanto, como os processadores têm apenas informação sobre a carga local, o balanceamento da carga pode não ser tão bom quanto nos algoritmos centralizados.

Os algoritmos semi-distribuídos dividem os processadores em regiões e dentro de cada região um algoritmo centralizado distribui a carga entre os processadores. Um mecanismo de mais alto nível distribui a carga entre as regiões.

Em diversos tipos de aplicações, os nodos representam coordenadas em um espaço bi- ou tridimensional e as arestas representam a comunicação ou dependência entre nodos vizinhos. Exemplos de tais aplicações incluem simulações realizadas empregando malhas de elementos finitos ou grades de diferenças finitas.

Para paralelizar a aplicação em uma coleção de processadores, o domínio computacional deve ser particionado entre os processadores. O particionamento deve ser feito de forma a atender três requisitos básicos:

1. A carga de trabalho deve ser bem balanceada entre os processadores;
2. A comunicação entre os processadores deve ser minimizada;
3. O tempo de execução do algoritmo de particionamento deve ser tal que o ganho obtido com seu uso seja maior que o custo de execução do algoritmo.

Em arquiteturas heterogêneas, onde diferentes nodos podem ter capacidades de processamento diferentes, a distribuição da carga entre os nodos deve levar em conta a capacidade de cada nó. Nodos com capacidades maiores devem receber cargas maiores de processamento. Isto deve ser levado em conta na distribuição inicial do domínio, através de um modelo do sistema (rede de interconexão e nodos) e durante a execução, quando podem ser utilizados mecanismos para verificar a ocupação dos nodos.

O problema de particionamento de domínio pode ser modelado como um problema de particionamento de grafos. Neste tipo de aplicação, os vértices do grafo representam as células do domínio e as arestas representam os dados trocados entre os subdomínios. A comunicação total entre os subdomínios é representada, então, pelo total de arestas entre os subdomínios (*edge-cut*). Na próxima seção será formalizado o problema de particionamento de domínio.

2.1 Particionamento de Domínio visto como particionamento de grafos

Um sistema paralelo com P processadores pode ser modelado como um grafo H=(U,F) com nodos U={0, ..., P-1} e arestas $F \subseteq U$ x U, chamado aqui de grafo de processadores. Para se considerar sistemas heterogêneos, pode-se associar pesos aos nodos e arestas.

Uma aplicação paralela pode ser modelada como um grafo G=(V,E,ρ, σ) com nodos V = {0, ..., N-1}, arestas $E \subseteq V$xV, pesos dos nodos $\rho : V \rightarrow \Re$ e pesos das arestas $\sigma : E \rightarrow \Re$. O significado dos nodos e arestas difere para cada aplicação. Por exemplo, nodos podem descrever processos ou dados, arestas podem significar dependências de dados ou necessidades de comunicação.

Usando este modelo, balanceamento de carga pode ser visto como um problema de mapeamento de grafos. A tarefa é encontrar um mapeamento $\pi : G \rightarrow H$ do grafo da aplicação para o grafo de processadores minimizando o critério de custo. Normalmente, busca-se um mapeamento em que a carga seja distribuída proporcionalmente à capacidade de cada processador, ao mesmo tempo em que se minimiza o corte de arestas, ou seja, o número de arestas entre os processadores.

Há outras métricas, além do corte de arestas, para a avaliação da qualidade da partição. Se as mensagens trocadas entre os processadores forem mensagens curtas, o tempo de latência será responsável pela maior parte do tempo de comunicação. Neste caso o número de mensagens trocadas com os processadores vizinhos, que corresponde no grafo ao número de subgrafos vizinhos, pode ser mais significativo que a quantidade de dados enviados e a minimização do número de subdomínios vizinhos pode ser uma métrica mais eficiente que a minimização de arestas de corte (HENDRICKSON; LELAND, 1995). Outros critérios como minimização da relação entre área e perímetro dos subdomínios, e número de vértices na fronteira podem ser utilizados, mas sem dúvida o corte de arestas é a métrica mais aceita e utilizada nos pacotes de particionamento.

Algumas implementações de algoritmos de particionamento, entre elas, a maioria das bibliotecas de particionamento descritas na seção 2.3, consideram que uma certa quantidade de desbalanceamento pode resultar em partições melhores, onde o ganho obtido na redução da comunicação compensa o desbalanceamento. O desbalanceamento é definido como o tamanho do maior subdomínio, S' = max $\|S_p\|$, dividido pelo tamanho ótimo de subdomínio S_{opt}. Se for permitido um desbalanceamento de $x\%$ então o problema de particionamento se torna: encontrar uma partição de G tal que o corte de arestas é minimizado sujeito à restrição $\|S_p\| \leq S_{opt}.(100 + x)/100$. O programa `kmetis`, presente na biblioteca METIS, descrito na seção 2.3.1, permite um desbalanceamento de até 3%.

Dependendo da aplicação o grafo G pode ser estático ou dinâmico, isto é, a carga computacional dos nodos da aplicação pode ou não ser alterada durante a execução. O grafo de processadores normalmente é considerado estático.

Aplicações estáticas devem ser mapeadas apenas uma vez no grafo de processadores e não são alteradas durante a execução. Assim, o problema de balanceamento de carga se reduz a um problema clássico de mapeamento onde um grafo deve ser projetado em outro.

Problemas de balanceamento dinâmico de carga ocorrem se o grafo da aplicação é alterado durante a execução. Arestas e vértices podem ser inseridos ou eliminados, ou os pesos podem ser alterados. Isto pode ocorrer, por exemplo, em uma simulação de corpos de água que considere o alagamento e secamento de regiões, ou no refinamento dinâmico de malhas, onde vértices e arestas são acrescentados durante a execução.

2.2 Métodos de Particionamento de Grafos

O problema de particionamento de grafos é o de dividir um grafo em subgrafos de forma que o número de arestas entre os subgrafos seja o menor possível. O número de subgrafos pode ser igual ao número de processadores, quando cada subgrafo será mapeado em um processador, ou maior que o número de processadores, situação em que cada processador poderá receber mais de um subgrafo.

Este problema, de dividir um grafo em k subgrafos com o mesmo número de vértices minimizando a comunicação é chamado de Problema de k-particionamento, e é um problema NP-difícil (GAREY; JOHNSON, ????), existindo apenas heurísticas que encontram uma boa aproximação e não a solução ótima. Algumas destas heurísticas são baseadas na bisecção recursiva do grafo. Para o problema de bisecção, muitas heurísticas eficientes existem que não encontram a solução ótima

(este problema também é NP-difícil) mas uma boa aproximação. Essas heurísticas diferem na qualidade da solução (número de arestas externas), no tempo de execução e nas informações adicionais necessárias.

Existem duas classes de heurísticas, dependendo das informações disponíveis sobre o grafo (DEMMEL, 1999b), que são as geométricas e as combinatórias.

O primeiro caso ocorre quando estão disponíveis informações sobre as coordenadas de cada vértice. Isto ocorre freqüentemente em grafos derivados da discretização de um domínio físico, como a aplicação abordada neste trabalho.

Para grafos com informações sobre as coordenadas, os algoritmos de particionamento são ditos geométricos, no sentido em que levam em consideração informações sobre a geometria do domínio. Alguns deles buscam dividir o domínio através de uma linha (para domínios bidimensionais) ou um plano (domínios tridimensionais), de modo que cada subdomínio fique com metade dos nodos. Estes algoritmos ignoram a colocação das arestas, considerando que os vértices próximos estão conectados por arestas. Os algoritmos de particionamento implementados no HIDRA fazem parte deste grupo.

O segundo tipo de grafos não possui coordenadas associadas aos vértices, isto é, não há identificação de um nodo com um ponto físico no espaço. Este tipo de grafo requer algoritmos combinatórios ao invés de geométricos. Um algoritmo simples de particionamento que não utiliza informação de coordenadas é o algoritmo de busca em amplitude (*breadth first search*), descrito na 2.2.5.

Heurísticas de particionamento normalmente são divididas em métodos globais e locais. Métodos globais são algumas vezes chamados de heurísticas de construção, uma vez que utilizam a descrição do grafo como entrada e geram uma partição. Métodos locais são chamados de heurísticas de melhoramento. Eles utilizam o grafo e uma partição como entrada e tentam melhorar o corte pelo rearranjo dos nodos. Nas seções a seguir, serão apresentadas algumas das principais heurísticas existentes, bem como o resultado da aplicação das mesmas ao HIDRA.

2.2.1 Métodos globais

A forma mais simples de particionamento de domínio e de menor custo computacional, aplicável a grades ortogonais resultantes de discretização por diferenças finitas, é o particionamento em faixas em qualquer uma das direções. Neste particionamento a fronteira entre os domínios pode ser uma linha reta (SSTRIP - *straight stripwise partitioning*), onde cada subdomínio recebe um certo número de linhas (ou colunas) inteiras, ou a linha (ou coluna) da fronteira pode ser dividida entre dois processadores vizinhos (STRIP - *stripwise partitioning*). O SSTRIP, apesar de ser de implementação mais simples e apresentar uma pequena vantagem do ponto de vista de comunicação, já que a fronteira entre os subdomínios é reta, não garante uma boa distribuição da carga entre os processadores dependendo fortemente do formato do domínio. O STRIP garante que a diferença no número de células entre dois subdomínios é no máximo de uma célula. Vollebregt (VOLLEBREGT et al., 1997) descreve uma aplicação onde foram utilizados o SSTRIP e o STRIP, além do particionamento em blocos (*Blockwise Partitioning*), bissecção recursiva ortogonal (ORB - *Orthogonal Recursive Bisection*) e a bissecção recursiva ortogonal reta (SORB - *Straight ORB*), descritas mais adiante. Uma descrição e análise dos principais algoritmos de particionamento pode ser encontrada em (ROEST, 1997).

Outra heurística de implementação simples, também apropriada a grades or-

togonais, é a distribuição por blocos (*Blockwise partitioning*) na qual a grade é dividida em blocos de mesmo tamanho. Isto pode levar a uma diferença grande de células entre os subdomínios, mas leva a fronteiras menores do que o particionamento por faixas e, conseqüentemente, a menos comunicação quando houver um número maior de processadores. Uma classe bastante popular de métodos de particionamento, aplicáveis tanto a grades ortogonais quanto a malhas não estruturadas, são os métodos de bissecção. Métodos de bissecção globais constroem uma bissecção válida de um grafo, isto é, geram uma divisão de nodos em dois subconjuntos de mesmo tamanho. Eles variam desde métodos muito simples que utilizam apenas numeração dos nodos até heurísticas muito sofisticadas que usam informações sobre a estrutura topológica do grafo. Suas necessidades de tempo de processamento e memória variam, bem como a qualidade das soluções geradas.

Métodos muito simples utilizam somente a numeração de nodos do grafo para dividi-los em duas partes. Eles geram partições de nodos com índices pares(ímpares), ou com índices maiores (menores) que a metade do número de nodos. Tais métodos são rápidos, mas a qualidade das soluções geradas depende fortemente da numeração dos nodos e, portanto, da forma como o grafo foi gerado. Normalmente estes métodos são pré-passos para sofisticadas heurísticas locais de melhoramento.

Se o grafo é uma malha, isto é, se ele está inserido em um espaço Euclidiano e coordenadas geométricas estão associadas a seus nodos, então esta informação extra pode ser utilizada para gerar partições ligeiramente melhores. Métodos simples de bissecção por coordenadas classificam os nodos de acordo com suas coordenadas x, y e z e então dividem o grafo nas direções x, y e z, dependendo de seu comprimento em qualquer uma das três direções. Nos métodos chamados de bissecção coordenada recursiva (*Recursive Coordinate Bisection* - RCB) a bissecção é aplicada recursivamente de forma balanceada ou não para cortá-lo em mais de duas partes. Quando a bissecção em um passo é feita numa direção ortogonal à bissecção do passo anterior, o método é conhecido por ORB (*Orthogonal Recursive Bisection*). No método ORB a bissecção pode ser feita dividindo a linha da fronteira entre dois processadores, o que garante subdomínios do mesmo tamanho, ou sem dividir nenhuma linha, cada processador recebendo apenas linhas inteiras, o que pode resultar em subdomínios de tamanhos diferentes. Neste caso o método é conhecido como SORB (*Straight ORB*)(VOLLEBREGT et al., 1997). O ORB pode também ser aplicado a malhas não estruturadas, como as geradas na discretização por elementos finitos (FOX; WILLIAMS; MESSINA, 1994). Neste caso, calcula-se o centro de massa dos elementos e a bissecção é feita por uma linha passando pelo centro de massa. Algumas variações do ORB, como o ORB-H (ORB hierárquico) permitem cortá-lo diretamente em um número maior de partições. Este método tem a vantagem de ser simples e ter baixo custo computacional (FOSTER, 1994). Um problema deste método é que ele não otimiza a comunicação, podendo levar a subdomínios com grandes fronteiras.

Simon (SIMON; TENG, 1993) apresenta uma análise da bissecção recursiva e apresenta alguns pontos contra (a partição obtida pode estar bem longe da partição ótima) e a favor (para alguns tipos de grafos a partição obtida é bem aceitável). Cabe observar que para o caso genérico de grafos não estruturados, o problema de encontrar uma bissecção ótima é NP-difícil (GAREY; JOHNSON, ????).

O método de bissecção coordenada recursiva foi implementado e aplicado ao HIDRA. A figura 2.1 mostra a divisão do domínio 4 processadores.

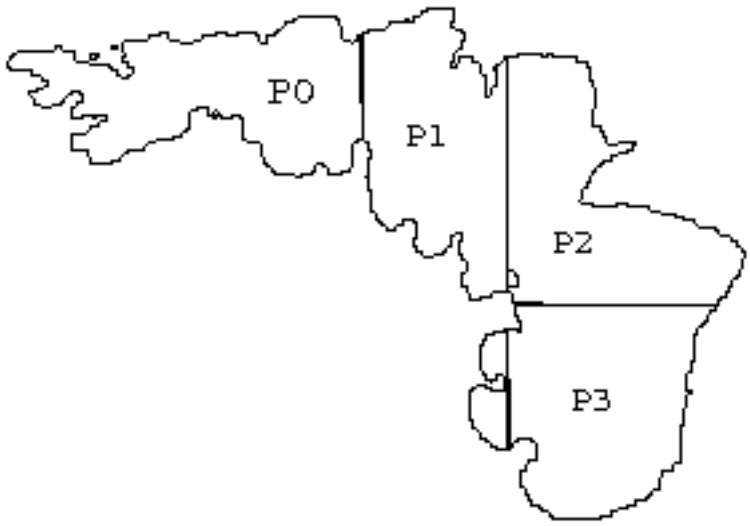


Figura 2.1: Divisão do domínio do Guaíba para 4 processadores por RCB

Uma variante da bissecção recursiva chamada bissecção recursiva desbalanceada (FOSTER, 1994) (*unbalanced recursive bisection*) tenta reduzir o custo da comunicação formando subdomínios com uma relação de aspecto melhor (relação entre a área do subdomínio e seu perímetro). Ao invés de automaticamente dividir o domínio pela metade, ele considera as P-1 partições obtidas formando subdomínios desbalanceados com 1/P e (P-1)/P da carga, com 2/P e (P-2)/P da carga e assim por diante, e escolhe a partição que minimiza a relação de aspecto. Este método aumenta o custo de calcular uma partição, mas reduz os custos de comunicação.

Um método ligeiramente mais elaborado que a bissecção coordenada recursiva inicialmente rotaciona o sistema de coordenadas de acordo com o eixo principal do grafo antes de utilizar bissecção coordenada. A heurística de bissecção inercial, descrita com mais detalhes na seção 2.2.2, calcula o centro de gravidade do grafo como a média ponderada das coordenadas de todos os nodos. Usando a norma euclidiana (ou L_2), ela então mede a distância de todos os nodos ao ponto central. As distâncias acumuladas nas direções x, y e z, e todas as combinações possíveis de x, y e z são agrupadas para formar uma matriz 3 x 3. O principal autovetor desta matriz determina o principal eixo do grafo. Ele é então particionado ortogonal a esse eixo.

Uma classe diferente de métodos não utiliza informação geométrica do grafo mas, ao invés, a conectividade. Os métodos ditos gulosos (*greedy*[1]) constroem uma bissecção partindo de um nodo (V_0) adicionando repetidamente o nodo que aumente o tamanho do menor corte (*cut size least*). Farhat (FARHAT, 1988) utiliza uma extensão desse método para particionar diretamente em k partições. Outras versões determinam diversos nodos iniciais e constroem cada parte em paralelo.

2.2.2 Particionamento inercial

O algoritmo de bissecção inercial (DEMMEL, 1999b) é simples: para um grafo com coordenadas 2D, ele escolhe uma linha tal que metade dos nodos fique de um

[1]Algoritmos gulosos é a denominação de uma família de algoritmos onde, para a obtenção do conjunto solução do problema, a cada passo busca-se o elemento que maximiza a solução parcial.

lado da linha e a outra metade fique do outro lado. Para um grafo com coordenadas 3D, ele escolhe um plano com a mesma propriedade. Uma vez escolhida uma linha L, a separação dos nodos em dois grupos pode ser feito como segue (o algoritmo é colocado para duas dimensões, apesar de ser extensível para qualquer número de dimensões).

1. Considere a linha reta L como dada por a(x-x')+b(y-y') = 0. É uma linha reta que passa por (x',y'), com inclinação -a/b. Pode-se assumir, sem perda de generalidade, que $a^2 + b^2 = 1$.

2. Para cada nodo $n_i = (x_i, y_i)$, computa-se uma coordenada $S_i = -b(x_i - x') + a(y_i - y')$. S_i é a distância de (x',y') à projeção de (x_i, y_i) na linha L.

3. Calcule a média S' dos $S_i's$.

4. Coloque os nodos (x_i, y_i) que satisfazem $S_i <= S'$ na partição N_1, e os nodos onde $S_i > S'$ na partição N_2.

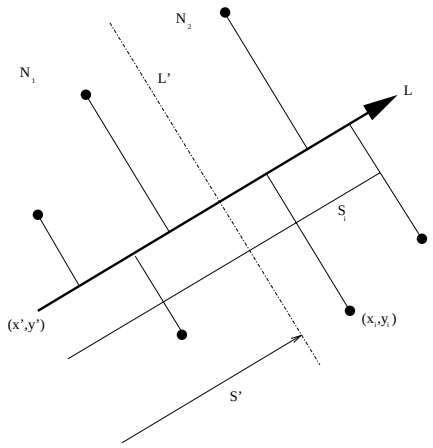


Figura 2.2: Método de bissecção inercial

O processo é mostrado na figura 2.2. A figura ilustra a bissecção de um grafo de 6 nodos. São mostradas a linha L e a projeção dos nodos sobre a linha L. N_1 e N_2 designam as duas partições, separadas pela linha L', definida como a linha ortogonal a L, com distância até (x', y') igual a S'. Nenhuma aresta é mostrada, uma vez que as mesmas não são usadas pelo algoritmo, apesar de estar claro que se apenas os vizinhos mais próximos estão conectados, então poucas arestas cruzarão a linha tracejada L' separando N_1 e N_2. Esse algoritmo pode ser implementado em tempo linearmente proporcional ao número de vértices do grafo.

Falta mostrar como se define a linha L. Intuitivamente, se os nodos estão localizados em uma região fina e comprida do plano, como no exemplo acima, é conveniente pegar L ao longo do maior eixo desta região. Em termos matemáticos, busca-se escolher uma linha tal que a soma dos quadrados das distâncias dos nodos às suas projeções sobre a linha L seja minimizada; isto é chamado de fazer um ajuste total de mínimos quadrados (*total least squares fit*) de uma linha aos nodos. Em termos físicos, se os nodos forem considerados unidades de massa, escolhe-se (x, y) como o eixo para o qual o momento de inércia dos nodos é minimizado, sendo esta a razão pela qual este método chama-se de particionamento inercial. Isto significa escolher

a, b, x' e y' de modo que $a^2 + b^2 = 1$, e o somatório dos quadrados das distâncias r_j dos nodos n_j à linha L seja minimizado. Este somatório é dado por

$$\begin{aligned}
\textstyle\sum_j r_j^2 &= \textstyle\sum_j [(x_j - x')^2 + (y_j - y')^2 - (-b(x_j - x') + a(y_j - y'))^2] \\
&= a^2 \textstyle\sum_j (x_j - x')^2 + 2ab \sum_j (x_j - x')(y_j - y') + b^2 \sum_j (y_j - y')^2 \\
&= a^2 X_1 + 2ab X_2 + b^2 X_3 \\
&= \begin{bmatrix} a & b \end{bmatrix} \begin{bmatrix} X_1 & X_2 \\ X_2 & X_3 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix}
\end{aligned}$$

Que pode ser minimizado pelas escolhas:
$(x', y') = (\sum_j x_j, \sum_j y_j)/N$, ou seja, é o centro de massa dos pontos (x_j, y_j)
(a, b) = autovetor unitário associado ao menor autovalor de $\begin{vmatrix} X_1 & X_2 \\ X_2 & X_3 \end{vmatrix}$

O caso 3D (e maiores) é similar, exceto que é obtida uma matriz 3x3. O software CHACO (HENDRICKSON; LELAND, 1995), descrito na seção 2.3.3, utiliza o método inercial para particionar o domínio em 2, 4 ou 8 subdomínios, usando 1, 3 ou 7 planos ortogonais ao eixo principal. Os subdomínios assim gerados são do tipo banda, o que pode acarretar um volume maior de comunicação do que o gerado, por exemplo, na aplicação recursiva de bissecção inercial. Entretanto, como o número de vizinhos de cada subdomínio é menor, o número de mensagens pode ser menor. Além disso, métodos inerciais multidimensionais podem ser mais rápidos que bissecção inercial por não ser necessário o cálculo de vários eixos e por não ocorrer o *overhead* da recursão.

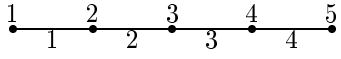
2.2.3 Bissecção espectral

Uma classe de métodos que geram partições melhores mas que têm maior custo computacional, são os chamados métodos de bissecção recursiva espectral (*Recursive Spectral Bisection*, RSB) ou bissecção recursiva por autovalor (*Eigenvalue Recursive Bisection*) (FOX; WILLIAMS; MESSINA, 1994)(SIMON, 1991)(LUCAS, 2000). O método básico, desenvolvido por Pothen, Simon e Liou é baseado em encontrar o autovetor correspondente ao segundo menor autovalor da matriz Laplaciana do grafo, chamado vetor de Fiedler, e utilizar este autovetor como separador.

Para poder apresentar o método de bissecção espectral são necessárias algumas definições. A matriz de incidências In(G) de um grafo G(N,E) é uma matriz $\|N\|$ x $\|E\|$, com uma linha para cada nodo e uma coluna para cada aresta. Para cada aresta $e(i, j)$, toda a coluna e é zerada, com exceção das linhas i e j, que possuem respectivamente 1 e -1. O Laplaciano (ou matriz Laplaciana) L(G) de um grafo G(N,E) é uma matriz simétrica $\|N\| x \|N\|$, com uma linha e uma coluna para cada nodo. Ele é definido por

- L(G) (i,i) = grau do nodo I se i=j (número de arestas incidentes)
- L(G) (i,j) = -1 se i≠j e existe a aresta (i,j)
- L(G) (i,j) = 0, caso contrário

As figuras 2.3a, 2.3b e 2.3c mostram o grafo de uma malha unidimensional, a matriz de incidência e a matriz Laplaciana associada. As figuras 2.4a, 2.4b e 2.4c mostram o grafo, a matriz de incidências e a matriz Laplaciana para uma malha 2-D.

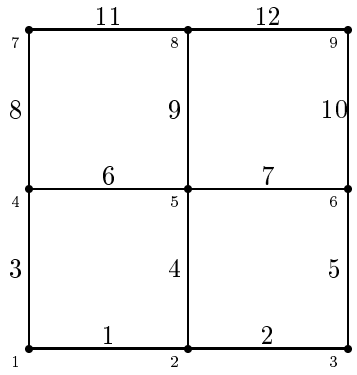


	1	2	3	4
1	-1			
2	1	-1		
3		1	-1	
4			1	-1
5				1

	1	2	3	4	5
1	1	-1			
2	-1	2	-1		
3		-1	2	-1	
4			-1	2	-1
5				-1	1

(a) (b) (c)

Figura 2.3: (a) Grafo de malha 1-D (b) Matriz de Incidência (c) Laplaciano



	1	2	3	4	5	6	7	8	9	10	11	12
1	-1		1									
2	1	-1		1								
3		1			1							
4			-1			-1		1				
5				-1		1	-1		1			
6					-1		1			1		
7								-1			-1	
8									-1		1	-1
9										-1		1

(a) (b)

	1	2	3	4	5	6	7	8	9
1	2	-1		-1					
2	-1	3	-1		-1				
3		-1	2			-1			
4	-1			3	-1		-1		
5		-1		-1	4	-1		-1	
6			-1		-1	3			-1
7				-1			2	-1	
8					-1		-1	3	-1
9						-1		-1	2

(c)

Figura 2.4: (a) Grafo de malha 2-D (b) Matriz de Incidência (c) Laplaciano

Dadas essas definições, o algoritmo de bissecção espectral é mostrado na figura 2.5.

O cálculo de v_2 e λ_2 de L(G) pode ser feito utilizando o procedimento de Lanczos mostrado na figura 2.6 (SJöSTROM, 96), o qual, para uma dada matriz simétrica A(n,n) (como L(G)), calcula de forma iterativa uma matriz tridiagonal simétrica T(k,k), cujos autovalores convergem para os autovalores de A. Um aspecto importante no procedimento de Lanczos é que os autovalores extremos de A aparecem bem antes da tridiagonalização estar concluída, o que torna o processo particularmente atraente para situações onde são necessários apenas alguns dos maiores ou menores

Calcule o autovetor v_2 correspondente ao segundo menor autovalor $\lambda_2(L(G))$
Para cada nodo n de G
Se $v_2(n) < 0$ então
coloque o nodo n na partição N-
Senão
ponha o nodo n na partição N+

Figura 2.5: Algoritmo de bissecção espectral

autovalores, como ocorre na bissecção espectral.

Dado r_0 arbitrário:
$\beta(0) = \|r\|$
j=0
repita
$j = j + 1$
$q_j = r_{j-1}/\beta(j-1)$ // escala um vetor
$r_j = A * q_j$ // produto matriz x vetor, o passo mais caro
$r_j = r_j - \beta(j-1) * q_{j-1}$ // escalar * vetor + vetor
$\alpha(j) = q_j^T * r_j$ // produto escalar
$r_j = r_j - \alpha(j) * q_j$ // escalar * vetor + vetor
$\beta(j) = \|r_j\|$ // calcula norma do vetor
até a convergência

Figura 2.6: Procedimento de Lanczos

Os vetores q_j são chamados de vetores de Lanczos. Este procedimento gera vetores de Lanczos até que $\beta(j) = 0$. A matriz T tem a estrutura mostrada abaixo e os autovalores/autovetores de A são aproximados utilizando os de T, existindo diversos métodos eficientes para o cálculo de autovalores de matrizes tridiagonais simétricas como o método da bissecção (GOLUB; LOAN, 1996).

$$T = \begin{array}{|c|c|c|c|c|c|} \hline \alpha(1) & \beta(1) & & & & \\ \hline \beta(1) & \alpha(2) & \beta(2) & & & \\ \hline & \beta(2) & \alpha(3) & \beta(3) & & \\ \hline & & \ldots & \ldots & \ldots & \\ \hline & & & \beta(k-2) & \alpha(\text{k-1}) & \beta(k-1) \\ \hline & & & & \beta(k-1) & \alpha(\text{k}) \\ \hline \end{array}$$

Os vetores de Lanczos formam uma base ortogonal no subespaço gerado por $(q_1, Aq_1, ..., A^j q_1)$ e os autovetores aproximados se encontram nesse subespaço. Entretanto, devido aos arredondamentos existentes na aritmética de ponto flutuante, a ortogonalidade não é garantida por muitas iterações, havendo diversas formas de abordar o problema. No método conhecido como Lanczos com reortogonalização completa a ortogonalização de um vetor q_j é feita contra todos os vetores anteriores da base, o que torna necessário armazenar todos os vetores. Em problemas de grande dimensão a quantidade de memória pode inviabilizar o uso deste método.

Os métodos de ortogonalização seletiva (GOLUB; LOAN, 1996) e reortogonalização parcial utilizam critérios para detectar quando ocorre perda de ortogonalidade da base e executar uma reortogonalização. Uma forma de manter baixo o número de vetores de Lanczos é reinicializar o processo após um determinado número de iterações com um novo vetor inicial.

Driessche e Roose (DRIESSCHE; ROOSE, 1995) acrescentaram restrições aos grafos desenvolvendo uma variante do método de bissecção espectral, que chamaram de Algoritmo de Bissecção Espectral Aperfeiçoado (*Enhanced Spectral Bisection Algorithm*). Métodos espectrais normalmente encontram uma boa partição mas que falha nos detalhes. Por essa razão, normalmente são utilizados conjuntamente com métodos de refinamento local (HENDRICKSON; LELAND, 1995). A figura 2.7 mostra um particionamento para 2 processadores gerada pelo método de bissecção espectral para o HIDRA, no qual uma das partições é desconexa. A figura 2.8 mostra a partição após aplicado o refinamento Kernighan-Lin. Ambas as partições foram geradas pelo pacote de particionamento CHACO, descrito na seção 2.3.3.

Figura 2.7: Particionamento do domínio do Guaíba pelo método da bissecção espectral sem refinamento KL

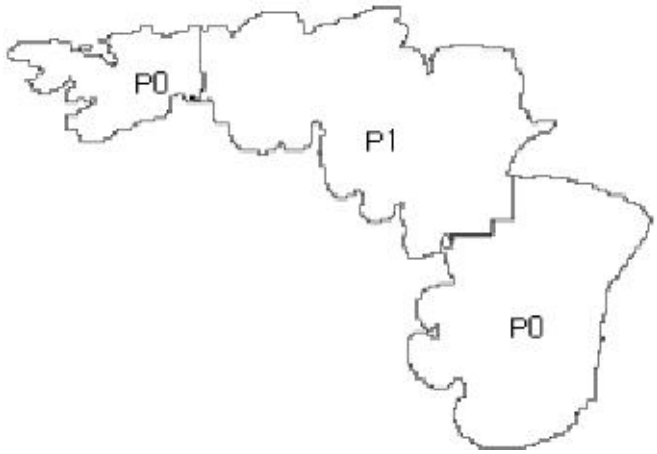


Figura 2.8: Particionamento do domínio do Guaíba pelo método da bissecção espectral com refinamento KL

2.2.4 Particionamento espectral multidimensional

Hendrickson e Leland (HENDRICKSON; LELAND, 1993) estenderam o método de bissecção espectral para permitir o particionamento em quatro (quadrissecção espectral) e oito (octassecção espectral) subdomínios, utilizando além do autovetor correspondente ao segundo menor autovalor, também os autovetores correspondentes

ao terceiro e quarto menores autovalores. Este procedimento reduz o número de níveis de bissecção reduzindo o custo computacional do cálculo dos autovetores.

2.2.5 Busca em amplitude (*Breadth First Search*)

Um algoritmo de particionamento muito simples é baseado na busca em amplitude (*breadth first search* - BFS) de um grafo. Dado um grafo conexo G=(V,E) e um vértice r em V, que será chamado de raiz, a busca em amplitude produz um subgrafo T de G (com todos os vértices e um subconjunto das arestas), onde T é uma árvore com raiz r. Além disso, ele associa um nível a cada vértice v, que é o número de arestas no caminho de r a v em T.

Existem diversos algoritmos de busca em amplitude, sendo os mais conhecidos os de Moore e o de Dijkstra para grafos valorados. A figura 2.9 mostra um grafo e a árvore gerada para o mesmo por uma busca em amplitude. Neste exemplo, os vértices que serão visitados a partir de um vértice v são selecionados a partir do vértice imediatamente abaixo, e em sentido horário. Na figura 2.9 à esquerda é assinalado o vértice escolhido como raiz e na figura 2.9 à direita os números indicam a ordem em que os vértices são visitados. A partir do vértice r são visitados os vértices 1, 2, 3 e 4, a partir do vértice 1 são visitados os vértices 5,6 e 7 e assim por diante. A árvore gerada tem 5 níveis, sendo o primeiro nível o da raiz, o segundo nível contém os vértices 1, 2, 3 e 4, e o último nível contém somente os vértices 18 e 19.

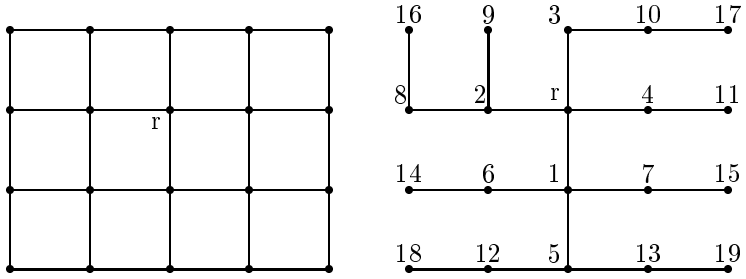


Figura 2.9: Grafo e árvore gerada por busca em amplitude

O fato mais importante sobre uma árvore BFS é que no grafo que originou a árvore não há arestas conectando nodos separados por mais de um nível. Em outras palavras, todas as arestas do grafo original são entre nodos do mesmo nível ou níveis adjacentes. A razão é a seguinte: Suponha que uma aresta conecta n_1 e n_2 e que (sem perda de generalidade) n_1 é visitado em primeiro lugar pelo algoritmo. Então o nível de n_2 é pelo menos tão grande quanto o nível de n_1. O nível de n_2 pode ser no máximo um a mais que o nível de n_1, porque este será visitado como filho de n_2. Um algoritmo baseado na busca em amplitude é o algoritmo proposto por Gibbs, Poole e Stockmeyer e utilizado no software SCOTCH. Neste algoritmo um vértice do grafo é escolhido aleatoriamente como raiz e o grafo é dividido em níveis por uma busca em amplitude. Um vértice do último nível é selecionado como nova raiz e o processo é repetido. Isto é feito até que o número de níveis da árvore não seja mais aumentado. Este algoritmo se baseia no pressuposto de que aumentando o número de níveis da árvore, cada nível terá menos vértices e assim o corte de arestas entre os diferentes níveis de vértices será menor.

2.2.6 Curvas de preenchimento de espaço

Uma abordagem que vem sendo bastante pesquisada recentemente para o particionamento de domínio é a utilização de curvas de preenchimento de espaço (*space-filling curves*) para particionamento de grafos multidimensionais (ALURU; SEVILGEN, 1997). Em 1890, Peano publicou a primeira descrição de uma curva de preenchimento de espaço. Um ano depois, Hilbert publicou a descrição de uma outra curva, talvez a mais simples de descrever. Hilbert dividiu um quadrado em quatro partes, e conectou seus quatro centros como mostrado na figura 2.10, para obter um mapeamento do intervalo unitário unidimensional para o quadrado.

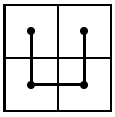


Figura 2.10: Configuração inicial da curva de Hilbert

A cada passo, então, o quadrado é dividido em quatro quadrados menores, a curva é reduzida para a metade de seu tamanho original e replicada nos quatro quadrados menores, sendo que nos quadrados superiores esquerdo e direito, a curva é rebatida em relação à diagonal. Após isso, as quatro curvas menores tem suas extremidades conectadas. As curvas geradas para três níveis podem ser vistas na figura 2.11.

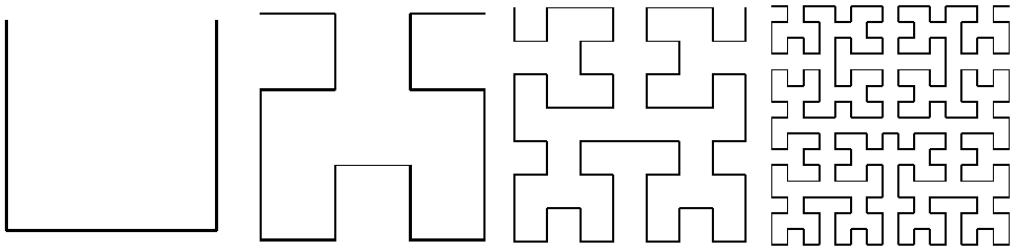


Figura 2.11: Curvas de Hilbert para três níveis

A cada passo o comprimento da curva vai ficando maior, mas como o algoritmo descrito pode ser repetido qualquer número de níveis, ele nunca termina. Assim, a curva obtida nunca preenche completamente o quadrado, assim como nunca ocorre um cruzamento entre segmentos da curva. O que preenche o quadrado é a curva limite desse algoritmo (que nunca é alcançada). Pode-se observar também que a curva obtida por esse procedimento pode chegar tão próximo de qualquer ponto do quadrado quanto se queira.

Uma curva pode ser vista como a imagem de um mapeamento contínuo do intervalo unitário $\mathbf{I} = [\mathbf{0}, \mathbf{1}]$ para o espaço multidimensional de n dimensões $\mathbf{R^n}$. Uma curva é dita de preenchimento de espaço (*space-filling*) se ela preenche todo um domínio. Há diversas curvas clássicas como Hilbert, Peano e Lebesgue, dentre as quais a mais utilizada no problema de particionamento de domínio é a curva de Hilbert, descrita no parágrafo anterior, devido à simplicidade de geração e boas características de localidade.

A curva de preenchimento de espaço pode ser utilizada também para o mapeamento inverso de um domínio multi-dimensional para o intervalo unitário I. A idéia

básica da utilização de curvas de preenchimento de espaço em particionamento de domínio é mapear dados multidimensionais para uma única dimensão onde o particionamento seja trivial. É uma característica da curva de Hilbert que elementos que são vizinhos no espaço unidimensional também serão no espaço multi-dimensional, ou seja, o mapeamento preserva a informação de proximidade no espaço multidimensional, o que minimiza custos de comunicação. Outras curvas, como a de Gray ou a curva Z, descritas em (ALURU; SEVILGEN, 1997) não apresentam esta característica. Parashar (PARASHAR; BROWNE, 1995) utilizou esta técnica para refinamento adaptativo de malhas. Zumbusch (ZUMBUSCH, 2001) utiliza esta técnica para particionamento de uma grade adaptativa resultante da aplicação de um método multinível à equação de Poisson.

Um particionador baseado em uma função adaptada de (PILKINGTON, 1996) foi implementado e aplicada sobre o domínio do Guaíba. Os testes realizados entretando, não resultaram em partições de qualidade, havendo um grande espaço para melhora na partição através do uso de métodos locais. Por essa razão não se deu continuidade à pesquisa com as mesmas, sendo utilizados métodos mais citados na literatura.

2.2.7 Outras abordagens para o problema de particionamento

Além dos algoritmos vistos nesse capítulo, que são os mais citados na literatura, outras abordagens também são utilizadas para particionamento de grafos (BHARGAVA et al., 1993). Entre essas abordagens podem ser citadas a utilização de algoritmos genéticos (FERENCZ et al., 1999), *simulated annealing* e redes neurais (FOX; WILLIAMS; MESSINA, 1994). Estas abordagens não serão discutidas nesse trabalho por serem pouco tratadas na literatura em comparação com os outros métodos.

2.2.8 Métodos multinível

Uma forma de reduzir o custo da bissecção de um grafo é a utilização de métodos multinível (SCHLOEGEL; KARYPIS; KUMAR, 2000). Em um algoritmo multinível, inicialmente o grafo é reduzido a um grafo menor através da contração de arestas (agrupamento de vértices) até que o número de vértices do grafo seja um múltiplo pequeno do número buscado de partições. O grafo menor é particionado utilizando alguma técnica como bissecção inercial, espectral ou Kernighan-Lin e, após, o grafo é descontraído. Para melhorar a qualidade do corte, podem ser usadas técnicas locais após cada passo da descontração do grafo.

A função mostrada na figura 2.12, extraída de (DEMMEL, 1999b), descreve o funcionamento de métodos multinível. Ela consiste de uma rotina de particionamento recursivo que recebe o conjunto de vértices e arestas que definem um grafo e retorna uma bissecção do mesmo :

A contração do grafo normalmente é baseada no conceito de conjunto independente de arestas. Um conjunto independente de arestas (*matching*) é um conjunto de arestas no qual não há duas arestas incidentes no mesmo vértice. Para contrair o grafo seleciona-se um conjunto independente de arestas e os vértices v_1 e v_2 adjacentes a cada aresta (v_1,v_2) pertencente ao conjunto são condensados em um único vértice v, com um peso igual à soma dos pesos de v_1 e v_2. Aqueles vértices que não são adjacentes a nenhuma aresta do conjunto são copiados para o grafo reduzido. Uma vez que o objetivo da contração é reduzir o grafo o máximo possível, o

```
Função Particiona_Recursivo( N, E ) : (N+,N-)

    Se |N| é pequeno então
        Particiona G=( N, E ) diretamente para obter N = N + ∪N−
        Retorna ( N+, N- )
     senão
        Calcula uma aproximação menor G_c = ( N_c, E_c )
        ( N_c+ , N_c- ) = Particiona_Recursivo( N_c, E_c )
        Expande ( N_c+, N_c- ) para uma partição ( N+, N- )
        Melhora a partição ( N+, N- )
        Retorna ( N+, N- )
    Fim se
Fim Função
```

Figura 2.12: Algoritmo de particionamento multinível

conjunto de arestas deve conter o maior número possível de arestas. Um conjunto maximal independente de arestas é um conjunto independente de arestas no qual não pode ser inserida mais nenhuma aresta sem quebrar o critério de não adjacência. Um conjunto maximal independente de arestas que tenha o maior número de arestas possível para um conjunto maximal de arestas é chamado de conjunto máximo independente de arestas. Como o custo computacional de encontrar um conjunto máximo independente de arestas é alto (o melhor algoritmo conhecido, devido a Edmonds é de complexidade $O(n^4)$ (GALIL, 1986)), utiliza-se conjuntos independentes maximais, não necessariamente máximos, mas que chegam a um bom resultado. Diversas abordagens podem ser utilizadas para selecionar as arestas que farão parte do conjunto de arestas a serem contraídas. Karypis e Kumar (KARYPIS; KUMAR, 1998a) apresentam 4 abordagens, todas elas de custo computacional linearmente proporcional ao número de arestas:

1. Seleção aleatória (*random matching*) - Os vértices são visitados em qualquer ordem. Para cada vértice u que ainda não foi associado a nenhum outro, é escolhido, entre os vértices adjacente a u, um que ainda não tenha sido associado a nenhum outro. Se tal vértice v existe, a aresta (u,v) é incluída no conjunto de arestas e os vértices u e v são marcados como já associados. Se não há vértice adjacente não associado, u permanece não associado no grafo reduzido.

2. Seleção das arestas de maior peso (*heavy edge matching*) - Considere-se um grafo $G_i = (V_i, E_i)$, um conjunto de arestas M_i utilizado para reduzir G_i e o grafo reduzido $G_{i+1} = (V_{i+1}, E_{i+1})$ induzido por M_i. Considere-se um conjunto de arestas A e uma função W(A) igual à soma dos pesos das arestas de A. Pode ser mostrado que $W(E_{i+1}) = W(E_i) - W(M_i)$, ou seja, a cada passo da redução do grafo, o peso total das arestas do grafo é reduzido de um valor igual à soma dos pesos do conjunto de arestas que serão contraídas. O algoritmo de seleção de arestas de maior peso tenta reduzir o peso total das arestas do grafo, selecionando para o conjunto de arestas independentes, arestas de maior peso. Considera-se que um grafo de menor peso total de arestas terá também um corte mínimo de arestas menor. O algoritmo seleciona aleatoriamente vértices entre os que ainda não foram associados e, para cada vértice u escolhido, a aresta (u,v) de maior peso incidente a ele e a um vértice v não associado é adicionada ao conjunto e os vértices u e v são marcados como associados.

3. Seleção das arestas de menor peso (*light edge matching*) - O algoritmo é praticamente o mesmo da heurística anterior. A diferença é que para cada vértice não associado é selecionada a aresta de menor peso entre todas as arestas possíveis. Isto resulta em grafos com um grau médio maior (o grau de um vértice é o número de arestas adjacentes a ele), o que pode ser útil para algumas heurísticas como a Kernighan-Lin.

4. Seleção de arestas de maior clique (*heavy clique matching*) - O algoritmo é semelhante à seleção de arestas de maior peso. A diferença é que nesta heurística é levado em conta para a seleção de uma aresta (u,v) não somente o peso da aresta (u,v) (ou das arestas que foram contraídas na aresta (u,v)), mas também o peso das arestas que foram contraídas no vértice u e no vértice v.

Dado um conjunto maximal independente de arestas E_m, o conjunto de arestas E_c do grafo contraído $G_c = (V_c, E_c)$ pode ser calculado pelo algoritmo mostrado na figura 2.13:

```
Para r = 1 até |E_m| /* Para cada nodo em N_c */
    Seja (i,j) a aresta em E_m correspondente ao nodo r
    Para cada outra aresta e=(i,k) em E incidente a i
        Seja e_k a aresta em E_m incidente a k, e r_k o nodo correspondente em N_c
            Adicione a aresta (r,r_k) a E_c
    Fim Para
    Para cada outra aresta e=(j,k) em E incidente a j
        Seja e_k a aresta em E_m incidente a k, e r_k o nodo correspondente em N_c
        Adicione a aresta (r,r_k) a E_c
    Fim Para
Fim para
Se existem arestas múltiplas entre pares de nodos de N_c então
    agrupe-as em arestas simples
```

Figura 2.13: Algoritmo de contração de arestas

A contração do grafo G para o grafo G_c é feita iterativamente ou recursivamente até que o tamanho de G_c seja menor que um determinado valor (normalmente algumas centenas de vértices). Ao chegar a esse ponto é aplicado um algoritmo de particionamento (espectral, inercial, Kernighan-Lin) e o grafo é expandido até o tamanho original, podendo ser refinado por algum algoritmo local a cada passo da expansão. No método espectral multinível (DEMMEL, 1999b), após o grafo G ser reduzido a G_c e particionado pelo método espectral, o autovetor utilizado na separação é interpolado a cada passo da expansão do grafo G_c para o grafo G_{c-1} e o autovetor resultante dessa interpolação é refinado. Nesse refinamento pode ser utilizado o método de Lanczos, ou o método chamado Iteração do Quociente de Rayleigh / SYMMLQ, que apresenta convergência mais rápida (GOLUB; LOAN, 1996)(DEMMEL, 1999b), que é um dos métodos utilizados no software de particionamento CHACO, descrito na seção 2.3.3.

2.2.9 Métodos locais

Apesar de alguns dos métodos globais produzirem bons cortes, freqüentemente há um potencial grande de melhora através de rearranjos locais. Heurísticas locais determinam conjuntos de nodos de mesmo tamanho que podem ser trocados entre as partes de modo que o tamanho do corte decresça. Estes métodos, normalmente

baseados na abordagem gulosa (*greedy*), diferem na forma como esses conjuntos são encontrados.

A forma mais simples de algoritmo guloso parte de uma partição inicial (A,B) dos vértices e calcula a cada passo uma nova partição através da troca de pares de vértices buscando maximizar o ganho. Isto é feito até que não haja mais trocas que resultem em ganho maior que zero. Há diversos algoritmos diferindo na forma como são escolhidos os vértices a serem trocados.

Uma simples busca local pode tentar trocar pares de nodos $a \in A, b \in B$ até que não seja possível mais nenhuma melhora. Este método é rápido mas as melhoras são limitadas devido à existência de mínimos locais, isto é, partições onde todas as trocas possíveis de nodos incrementariam o corte.

Para determinar de forma eficiente o custo de uma troca, pode-se armazenar para cada vértice $a \in A$ seu custo interno $I(a)$, que é o número de arestas (a, a') tal que $a' \in A$, e seu custo externo $E(a)$, que é o número de arestas (a, b) tal que $b \in B$. Isto pode ser feito em tempo $O(m + n)$, onde m é o número de arestas do grafo e n é o número de vértices e após a troca de a e b, a atualização pode ser feita em tempo $O(grau(a) + grau(b))$. O ganho na troca de a e b (número de arestas de corte antes da troca menos o número de arestas de corte após a troca) é dado por $E(a) - I(a) + E(b) - I(b) - 2w(a, b)$, onde $w(a, b) = 1$ se existe a aresta (a, b), e $w(a, b) = 0$, caso contrário. Pode-se então determinar o melhor par para trocar em tempo $O(n^2)$ simplesmente executando a troca para todos os $(n/2)^2$ pares candidatos, guardando os de maior ganho para uma escolha aleatória ao final.

2.2.10 A heurística Kernighan-Lin

A heurística Kernighan-Lin (algoritmo KL) (DEMMEL, 1999b) usa a mesma operação básica, troca de pares de nodos, mas permite um aumento temporário no tamanho do corte, evitando, assim, mínimos locais. A versão original, publicada por B. Kernighan e S. Lin (KERNIGHAN; LIN, 1970), requer $O(N^3)$ operações por iteração. Uma versão mais eficiente mas bem mais complexa, de ordem $O(\|E\|)$, foi apresentada por C. Fiduccia e R. Mattheyses (FIDUCCIA; MATTHEYSES, 1982) e é descrita na seção 2.2.11.

O algoritmo de Kernighan-Lin é projetado para ser utilizado em grafos com pesos associados às arestas, mas pode ser igualmente utilizado para grafos não valorados, assumindo peso unitário para todas as arestas. Ele inicia com um grafo $G = (N, E)$ e uma partição inicial dos nodos $G = A \cup B$ tal que $\|A\| = \|B\|$. Seja $w(e) = w(i, j)$ o peso da aresta $e = (i, j)$, sendo igual a 0 se a aresta não existe. O objetivo é encontrar subconjuntos X em A e Y em B, de mesmo tamanho, tais que a troca de X e Y reduza o custo total das arestas que ligam A a B. Mais precisamente, $T = \sum_{[a \in A \ e \ b \in B]} w(a, b) =$ custo das arestas entre A e B, e busca-se X e Y tais que a nova partição $novo\_A = (A - X) \cup Y$ e $novo\_B = (B - Y) \cup X$ têm um custo menor novo_T. Para o cálculo de novo_T serão utilizadas as seguintes definições:

$E(a) =$ custo externo do vértice $a \in A = \sum_{[b \in B]} w(a, b)$
$I(a) =$ custo interno do vértice $a \in A = \sum_{[a' \in A, a' \neq a]} w(a, a')$
$D(a) =$ custo do vértice $a \in A = E(a) - I(a)$

e, de forma análoga,

$E(b) =$ custo externo de $b \in B = \sum_{[a \in A]} w(a,b)$
$I(b) =$ custo interno de $b \in B = \sum_{[b' \in B, b' \neq b]} w(b,b')$
$D(b) =$ custo de $b \in B = E(b) - I(b)$

Observa-se que com a troca de a em A por b em B, todas as arestas adjacentes a a que são internas à partição A passam a ser externas e vice-versa, assim como para b. Assim, a troca de a por b altera T para

$$novo\_T = T - (D(a) + D(b) - 2 * w(a,b)) = T - ganho(a,b)$$

onde $ganho(a,b) = D(a) + D(b) - 2 * w(a,b)$ mede a melhora da partição após a troca dos vértices em A pelos vértices em B. $D(a')$ e $D(b')$ são também alterados para:

$novo\_D(a') = D(a') + 2 * w(a',a) - 2 * w(a',b)$ para todo $a' \in A$, $a' \neq a$
$novo\_D(b') = D(b') + 2 * w(b',b) - 2 * w(b',a)$ para todo b' $\in B$, $b' \neq b$

O grafo na figura 2.14, no qual os vértices marcados com uma bolinha pertencem à partição A e os não marcados pertencem à partição B, será utilizado para ilustrar o ganho decorrente da troca dos vértices 1 e 2.

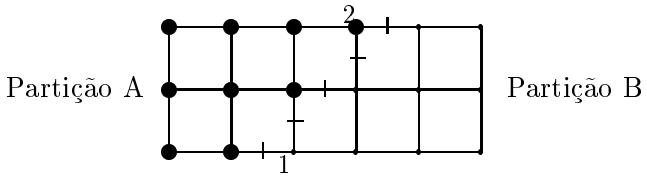


Figura 2.14: Ganho obtido com a troca de um par de vértices

Para a partição apresentada na figura, considerando um peso unitário para cada aresta, há 5 arestas entre as partições A e B (marcadas com um traço transversal) e, portanto, $T = 5$. O vértice 1, pertencente à partição B, está ligado a 2 vértices da partição A e a 1 vértice da partição B, portanto, $E(1) = 2$, $I(1) = 1$ e $D(1) = E(1) - I(1) = 1$. Para o vértice 2, $E(2) = 2$, $I(2) = 1$ e $D(2) = E(2) - I(2) = 1$. Como os vértices 1 e 2 não são adjacentes, $w(1,2) = 0$. Após a troca de 1 e 2, $novo\_T = T - (D(1) + D(2) - 2 * w(1,2)) = 5 - (1 + 1 - 2 * (0)) = 3$ e o $ganho(1,2) = 2$. Após essas definições é apresentado, na figura 2.15, o algoritmo Kernighan_Lin (os comentários mostram o custo de cada passo do algoritmo)

Em um passo do laço Repita este algoritmo computa $|N|/2$ possíveis pares de conjuntos A e B para trocar, $A = (a_1, ..., a_j)$ e $B = (b_1, ..., b_j)$, para j=1 to $|N|/2$. Esses conjuntos são escolhidos pelo método guloso, portanto a troca de a_i e b_i apresenta o maior ganho possível na troca de qualquer par de nodos. O melhor destes $|N|/2$ pares de conjuntos é escolhido, maximizando o ganho. Alguns ganhos podem ser negativos, se nenhum ganho pode ser obtido trocando apenas dois nodos. Entretanto, ganhos posteriores podem ser grandes e assim o algoritmo escapa de mínimos locais. O custo de cada passo é mostrado à direita do algoritmo. Uma vez que a busca do par não marcado de ganho máximo (passo 1 do Enquanto) é repetido $O(|N|)$ vezes, o custo de um passo do laço Repita é $O(|N|^3)$. O número de passos do laço Repita é difícil de prever. Testes empíricos de Kernighan e Lin em

```
Calcula T = custo da partição = A ∪ B /* custo = O(|N|²) */
Repita
    Calcula custos D(n) para todo n em N /* custo = O(|N|²) */
    Desmarca todos os nodos em G /* custo = O(|N|) */
    Enquanto há nodos desmarcados /* |N|/2 iterações */
        Encontra um par não marcado (a,b) maximizando ganho(a,b) /* custo = O(|N|²) */
        Marca a e b (mas não os troca) /* custo = O(1) */
        Atualiza D(n) para todos os n não marcados, como se a e b tivessem sido trocados
            /* custo = O(|N|) */
    Fim Enquanto
        /* Neste ponto, foi computada uma seqüência de pares (a_1, b_1),...,(a_k, b_k) e ganhos
           ganho(1),...,ganho(k) onde k = |N|/2, ordenados pela ordem em que foram marcados */
    Selecione j maximizando Ganho = Σ_{i=1...j} ganho(i)
        /* Ganho é a redução no custo resultante da troca (a_1,b_1), ...,(a_j,b_j) */
    Se Ganho > 0 então
        Atualize A = A - {a_1,...,a_j} ∪ {b_1,...,b_j} /* custo = O(|N|) */
        Atualize B = B - {b_1,...,b_j} ∪ {a_1,...,a_j} /* custo = O(|N|) */
        Atualize T = T - Ganho /* custo = O(1) */
    Fim se
Até Ganho ≤ 0
```

Figura 2.15: Algoritmo Kernighan-Lin

grafos pequenos ($|N| \leq 360$), mostraram convergência entre 2 e 4 passos. Para um grafo gerado aleatoriamente, a probabilidade de chegar ao ótimo em um passo reduz para $2^{|N|/30}$. Uma implementação paralela deste algoritmo foi apresentada por J. Gilbert e E. Zmijewski (GILBERT; ZMIJEWSKI, 1987). Mais recentemente outras melhoras significativas foram propostas por G. Karypis e V. Kumar em (KARYPIS; KUMAR, 1998a).

2.2.11 O algoritmo de Fiduccia-Mattheyses

O algoritmo de Fiduccia-Mattheyses é baseado no algoritmo de Kernighan-Lin. Ele consiste de uma série de passagens sobre todo o grafo nas quais vértices são movidos de uma partição para a outra. A seleção do vértice a ser movido é baseada no mesmo conceito de ganho do algoritmo de Kernighan-Lin. O vértice de maior ganho é aquele cuja movimentação irá reduzir ao máximo o corte de arestas. Movimentos que piorem a qualidade da partição são permitidos, o que possibilita ao algoritmo escapar de mínimos locais. Dentro de um passo um vértice pode ser movido apenas uma vez para evitar loops infinitos. A cada passo os vértices são divididos em dois grupos, o dos vértices livres, que ainda não foram movimentados, e o dos vértices bloqueados, que já foram movimentados. A estrutura básica de um passo é a seguinte (HENDRICKSON; KOLDA, 1998):

1. marque todos os vértices como livres;

2. para cada vértice, calcule o ganho associado à sua movimentação de uma partição para outra. Ganhos positivos reduzem o custo de uma solução. Ganhos negativos aumentam o custo;

3. entre os movimentos que melhoram o critério de balanceamento ou que ao menos não violam o critério de balanceamento, selecione o vértice livre que apresenta o maior ganho. Se não há mais vértices livres, termina o passo;

4. mova o vértice selecionado para a outra partição, marque-o como bloqueado e atualize os ganhos de todos seus vizinhos;

5. se esta é a melhor partição vista até o momento, salve-a;

6. vá para o passo 3.

A melhor solução encontrada em um passo é utilizada como solução inicial para o próximo passo. Isto é repetido até que ocorra um passo em que a solução não é melhorada. Fiduccia e Mattheyses observaram que o ganho obtido deve estar entre $2 * max$ e $-2 * max$, sendo max o grau do vértice de maior grau do algoritmo, o que permite agrupar os vértices pelo seu ganho em uma estrutura de dispersão (*hash*), permitindo atualizar todos os ganhos em tempo linear. Isto faz com que a execução de um passo tenha complexidade linear. Uma discussão detalhada das estruturas de dados pode ser encontrada em (FIDUCCIA; MATTHEYSES, 1982) e (SANTOS, 2001).

Embora os algoritmos Kernighan-Lin e Fidducia-Mattheyses consigam escapar de mínimos locais em algumas situações, essa capacidade é muito limitada. Assim, a qualidade da bissecção final obtida por esses algoritmos é muito dependente da qualidade da bissecção inicial. Técnicas têm sido desenvolvidas para melhorar esses algoritmos através da movimentação de conjuntos maiores de vértices, mas ao custo do aumento da complexidade. Os algoritmos Kernighan-Lin e Fidducia-Mattheyses tendem a ser mais efetivos quando o grau médio dos vértices do grafo é alto e quando as restrições de balanceamento são relaxadas, razão pela qual os pacotes de particionamento usualmente permitem definir qual o percentual de desbalanceamento permitido.

2.3 Pacotes de Software para Particionamento de Grafos

Existem diversos pacotes e bibliotecas de particionamento de domínio. Alguns dos pacotes mais conhecidos de particionamento de grafos são:

- MÉTIS, de Karypis e Kumar;
- JOSTLE, de Walshaw;
- CHACO, de Hendrickson e Leland;
- SCOTCH, de Pellegrini.

Nas próximas seções serão descritos brevemente os pacotes citados, bem como os algoritmos empregados em cada um. Cabe salientar que o SCOTCH diferencia-se dos outros por ser, mais do que um pacote para particionamento de grafos, um pacote para mapeamento de grafos, permitindo mapear o grafo de uma aplicação sobre o grafo de processadores, utilizando arestas e vértices valorados tanto no grafo da aplicação quanto no grafo dos processadores.

2.3.1 O software MÉTIS

O software MÉTIS é um pacote de particionamento de grafos e reordenamento de matrizes esparsas desenvolvido por George Karypis e Vipin Kumar (KARYPIS; KUMAR, 1998b). Ele pode ser utilizado como um programa de particionamento ou

como uma biblioteca de rotinas de particionamento de grafos, existindo duas versões do programa, utilizando diferentes algoritmos para o particionamento. O `pmetis` utiliza um algoritmo de particionamento baseado em bissecção recursiva multinível e o `kmetis` é baseado em um algoritmo de k-particionamento multinível, descrito em (KARYPIS; KUMAR, 1998c). O `pmetis` garante um balanceamento perfeito a cada nível da recursão, enquanto o `kmetis` permite um desbalanceamento de até 3%. O pacote MÉTIS está disponível eletronicamente através da Internet (KARYPIS; KUMAR, 1998b), não sendo necessária licença de uso.

2.3.2 O software JOSTLE

O software JOSTLE (WALSHAW, 2000), escrito por C. Walshaw, M. Cross, e M. Everett da Universidade de Greenwich utiliza, assim como o MÉTIS, um refinamento multinível. O grafo original sofre uma série de contrações gerando grafos cada vez menos refinados. Sobre o grafo menos refinado é gerada uma partição inicial que é repetidamente interpolada para o grafo mais refinado imediatamente superior onde sofre um processo de balanceamento. O algoritmo de refinamento é uma versão multidimensional do algoritmo de otimização iterativo de Kernighan-Lin, podendo utilizar também um algoritmo guloso, de menor custo computacional mas que resulta em um refinamento de pior qualidade.

O JOSTLE utiliza uma série de parâmetros que permitem definir, entre outras características:

- qual o desbalanceamento aceitável;
- até que ponto é feita a redução do grafo nos métodos multinível;
- se será aplicado um método global ou local;
- qual o método local utilizado.

O JOSTLE pode ser utilizado para refinar uma partição já existente utilizando, neste caso, apenas um método local de refinamento. Uma característica interessante do JOSTLE é o suporte para problemas multi-fase, onde o grafo representa diversas fases computacionais separadas por pontos de sincronização. Para isso, a cada vértice do grafo é associado um vetor especificando qual a contribuição daquele vértice em cada fase do processo. Por exemplo, se existem duas fases, um nodo que participa apenas da fase 1 teria um vetor $[w_1,0]$, enquanto que um vértice que contribuísse somente para a fase 2 teria um vetor $[0,w_2]$. Problemas multi-fase são discutidos na seção 2.4.

Além da versão seqüencial do JOSTLE, foi desenvolvida também uma versão paralela, chamado Pjostle, utilizando a biblioteca MPI. Ambas as versões estão disponíveis eletronicamente através da Internet (WALSHAW, 2000), sendo necessário o preenchimento de uma licença de uso. Entretanto, foi feito contato com o autor para obter uma licença de uso e após alguns meses veio a resposta, com a lista de plataformas disponíveis, da qual não constava o sistema operacional Linux.

2.3.3 O software CHACO

O software CHACO (HENDRICKSON; LELAND, 1995), desenvolvido por Bruce Hendrickson e Robert Leland, no Los Alamos National Labs, implementa diversos

algoritmos de particionamento. Entre eles o método inercial, espectral, Kernighan-Lin e métodos multinível. Cada um desses algoritmos pode particionar o grafo em 2, 4 ou 8 subgrafos em cada estágio da decomposição recursiva. O CHACO permite que a saída de qualquer método global de particionamento seja utilizada em qualquer dos métodos locais de refinamento. Além dos algoritmos citados, o CHACO utiliza três esquemas simplificados de particionamento. No particionamento linear, os vértices são distribuídos entre os subgrafos de acordo com sua numeração. Para um grafo não valorado com n vértices sendo dividido em p subgrafos, os primeiros n/p vértices seriam atribuídos ao primeiro subgrafo, os próximos n/p vértices seriam atribuídos ao segundo subgrafo e assim por diante. No particionamento randômico, vértices são atribuídos aleatoriamente a subgrafos mantendo o balanceamento. No particionamento espalhado (*scattered*), os vértices são atribuídos a subgrafos, cada vértice por sua vez sendo atribuído ao menor subgrafo existente até o momento.

Além do particionamento do grafo, o CHACO efetua o mapeamento do grafo particionado a arquiteturas hipercúbicas ou malhas 1-D, 2-D e 3-D, mas de forma bastante limitada, não permitindo a especificação do grafo de processadores como o SCOTCH.

A utilização do CHACO é baseada em um menu inicial onde é solicitado o algoritmo global e local a ser utilizado, número de subdomínios e parâmetros específicos do método de particionamento utilizado. Alguns parâmetros são buscados de um arquivo de configuração. Como resultado, além da partição resultante, o CHACO fornece uma série de informações como o tempo utilizado em operações de entrada e saída e particionamento, número de arestas de corte, número de vértices internos, número de vértices de fronteira e outras.

O pacote está disponível eletronicamente, sendo necessário solicitar uma licença de uso ao autor (HENDRICKSON; LELAND, 1995).

A figura 2.16 mostra o particionamento do Guaiba em 16 subdomínios utilizando o CHACO.

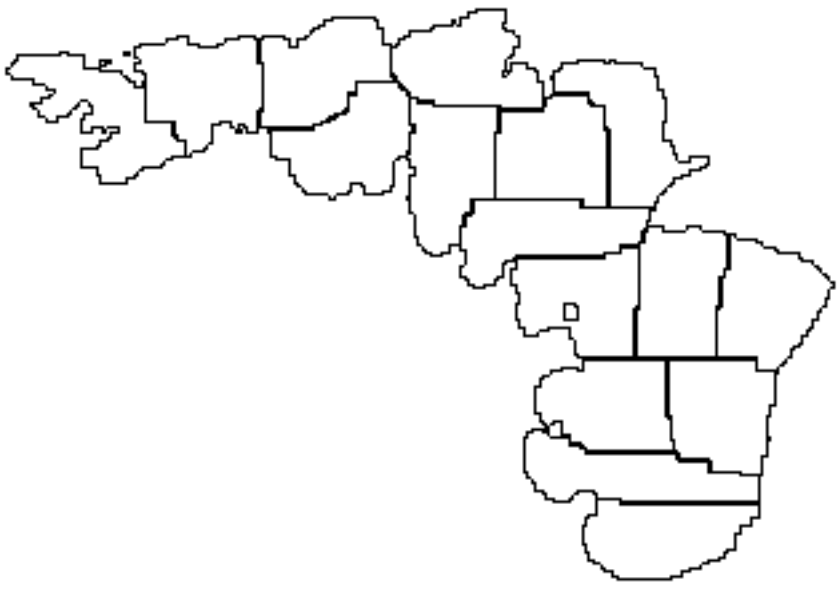


Figura 2.16: Particionamento do Guaiba 200 em 16 processos com o CHACO

2.3.4 O software SCOTCH

O SCOTCH (PELLEGRINI, 1996), desenvolvido por François Pellegrini no Laboratoire Bordelais de Recherche en Informatique (LaBRI), da Université Bordeaux I, é um pacote de particionamento e mapeamento de grafos, permitindo mapear o grafo de uma aplicação sobre o grafo de uma máquina paralela utilizando para isso um algoritmo de biparticionamento recursivo dual. Neste algoritmo a estrutura da máquina paralela sobre a qual vai rodar a aplicação é representada como um grafo valorado onde os pesos dos vértices representam a capacidade de processamento dos nodos da máquina e os pesos das arestas representam a capacidade de comunicação entre os nodos. O algoritmo utiliza uma abordagem de divisão e conquista para recursivamente alocar subconjuntos de processos a subconjuntos de processadores. Ele começa considerando um conjunto de processadores, também chamado de domínio, contendo todos os processadores da máquina alvo. A cada passo o algoritmo biparticiona um domínio ainda não processado em dois subdomínios disjuntos, e chama um algoritmo de biparticionamento de grafos para dividir o subconjunto de processos associados. O SCOTCH permite a utilização de diversos algoritmos para particionamento do grafo de processos, entre eles:

- Método de busca exaustiva através de *backtracking*. Esgota o espaço de busca de soluções gerando todos os mapeamentos possíveis. Devido ao custo computacional pode ser utilizado apenas em grafos pequenos;
- Método de Fiduccia-Mattheyses, descrito na seção 2.2.11 ;
- Método de Gibbs-Poole-Stockmeyer, descrito na seção 2.2.5;
- Método aleatório, onde são gerados aleatoriamente conjuntos de vértices e arestas. Utilizados para geração de dados para comparação estatística com outros métodos;

2.3.5 Utilização dos pacotes no HIDRA

O objetivo do estudo dos pacotes apresentados era avaliar a viabilidade de sua utilização nos modelos sendo desenvolvidos, bem como ter um termo de comparação para os algoritmos de particionamento implementados (basicamente particionamento por faixas e diversas versões de bissecção coordenada recursiva) em relação à qualidade do particionamento gerado. Os dois pacotes utilizados, dentre os que foram avaliados, foram o MÉTIS e o CHACO. O JOSTLE não foi utilizado pela dificuldade em se obter uma versão compatível com o ambiente e ferramentas utilizadas. O SCOTCH, por sua vez, não foi utilizado por não apresentar, para a aplicação sendo desenvolvida, vantagens em relação aos outros dois. A sua maior vantagem sobre os outros pacotes, que é a capacidade de efetuar o mapeamento do grafo da aplicação sobre o grafo de processadores não seria utilizada no HIDRA uma vez que o problema de mapeamento não foi abordado, já que o grafo do *cluster* utilizado é um grafo completo.

O pacote mais utilizado no modelo foi o MÉTIS. A razão para isso foi a facilidade de utilização do mesmo como uma biblioteca de rotinas de particionamento, o que não ocorre com os outros pacotes avaliados. Como ponto negativo do MÉTIS, ele não oferece muitas alternativas de algoritmos de particionamento, limitando-se a sugerir que para um número pequeno de partições (até 8 partições) seja usada a função

METIS_PartGraphRecursive e para um número maior de partições seja usada a função METIS_PartGraphKway.

Algumas partições geradas pelo MÉTIS eram passíveis de melhora no corte de arestas através do uso de um algoritmo de refinamento, o que levou a avaliar a possibilidade de utilizar o CHACO, que oferece um bom número de métodos de particionamento, incluindo refinamento local via Kernighan-Lin. O uso do CHACO como biblioteca de funções de particionamento não se revelou tão simples como o MÉTIS. A alternativa foi a utilização do CHACO a partir de linha de comando gerando arquivos com as partições, arquivos esses que eram lidos pelo modelo no início. As medidas de desempenho efetuadas com as partições geradas pelo CHACO, entretanto, não mostraram melhora significativa em relação ao MÉTIS ou ao RCB e o CHACO não foi adotado como particionador do modelo. Foram feitos diversos testes comparando o tempo de execução do HIDRA-3D utilizando o MÉTIS e o RCB. Os resultados dessa comparação podem ser vistos na seção 7.5.

2.4 Particionamento de domínio em problemas multi-fase

Uma classe particular de problemas de particionamento e balanceamento ocorre quando o fenômeno modelado é dividido em diferentes fases, separadas por um ponto de sincronização explícita, com diferentes distribuições de carga em cada fase. Walshaw (WALSHAW; CROSS; MCMANUS, 1999) chama essa classe de problemas de multi-fase, e a aplicação desenvolvida neste trabalho, um modelo de hidrodinâmica e transporte de substâncias, pode ser enquadrada nesta classe de problemas.

Uma abordagem possível para esse problema é associar a cada vértice do grafo do problema um vetor de m posições, sendo m o número de fases, onde cada posição do vetor contém o custo do processamento daquele vértice para aquela fase. Em fases onde um determinado vértice não necessite ser considerado, considera-se seu peso igual a zero. Assim, em um problema de duas fases, com pesos unitários para cada vértice, os vértices que participariam apenas na primeira fase teriam associado a eles o vetor [1 0], vértices que participassem apenas da segunda fase teriam associado a eles o vetor [0 1], e vértices que participassem de ambas as fases teriam associado o vetor [1 1].

Walshaw utiliza esse modelo de múltiplos pesos por vértice para solucionar problemas multi-fase. Em sua abordagem, os vértices são classificados de acordo com a fase em que participam pela primeira vez. Assim, vértices que participam da primeira fase são classificados como vértices do tipo 1, vértices que participam somente a partir da segunda fase são classificados como vértices do tipo 2 e assim por diante. O algoritmo de Walshaw basicamente efetua um particionamento por fases, particionando a cada fase apenas aqueles vértices que ainda não foram considerados nas fases anteriores. A cada nova fase, os vértices da fase anterior são condensados em um vértice representativo de cada subdomínio, de forma a reduzir o grafo e tornar mais eficiente o particionamento das fases subseqüentes.

Karypis (KARYPIS; KUMAR, 1998d) considera este tipo de problema como um problema de múltiplas restrições (*multi-constraint*), onde a cada fase deve ser resolvido um problema de minimização do corte de arestas mantendo o balanceamento da carga dentro de uma tolerância especificada. Esta tolerância pode ser representada por um vetor c_n, onde n é o número de fases do problema. Por exem-

plo, um vetor c=[1.05 1.5] significa que a primeira fase deve ter no máximo 5% de desbalanceamento e a segunda fase deve ter no máximo 50% de desbalanceamento.

2.4.1 O HIDRA visto como um modelo multi-fase

O HIDRA é um modelo de hidrodinâmica e transporte de substâncias. A hidrodinâmica e o transporte de substâncias são resolvidos em fases distintas, com resoluções diferentes e sobre malhas diferentes do domínio. Devido à natureza dos processos físico-químicos envolvidos no transporte de substâncias, para que os resultados obtidos pela simulação sejam de boa qualidade é necessário que a resolução utilizada seja da ordem de poucas dezenas de metros, enquanto que a resolução utilizada na hidrodinâmica é da ordem de algumas centenas de metros.

Outra característica do problema de transporte de substâncias, particularmente da simulação do transporte de coliformes como tratado nesse trabalho, é que o fenômeno não ocorre de maneira uniforme em todo o domínio. Nesse tipo de problema, normalmente existem alguns pontos emissores de poluentes, próximo aos quais ocorrem altos gradientes de concentração. À medida que os pontos considerados vão se afastando do emissor, os gradientes de concentração vão diminuindo, em decorrência da dispersão do poluente, resultado da difusão e advecção e, no caso de coliformes, do decaimento dos mesmos. A partir de um certo ponto o valor numérico das concentrações envolvidas na simulação é tão baixo que não tem efeito notável na simulação e pode não ser levado em conta.

No HIDRA é utilizada uma grade retangular para a hidrodinâmica. A simulação do transporte de substâncias é efetuada sobre uma parte refinada do domínio da hidrodinâmica, utilizando informações resultantes dessa última, como as velocidades e nível em cada célula. Para a obtenção da resolução maior, aquelas células do domínio da hidrodinâmica onde ocorre transporte de substâncias são refinadas gerando uma outra grade sobreposta à da hidrodinâmica, como mostrado na figura 2.17, onde cada célula da hidrodinâmica foi refinada por um fator igual a 3. As duas regiões mais escuras na figura são as áreas onde a malha foi refinada. A figura 2.18 mostra em detalhes a região do domínio próxima à área refinada.

Figura 2.17: Malha da hidrodinâmica e do transporte de substâncias

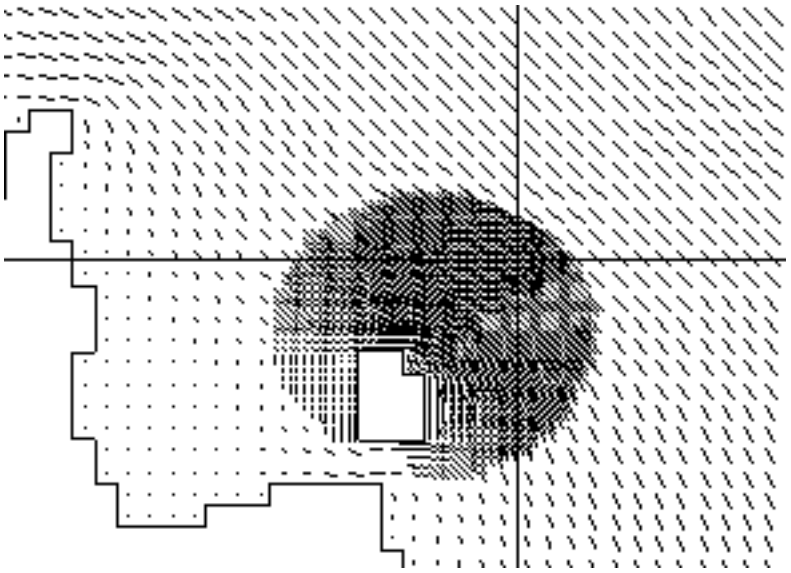


Figura 2.18: Vista aproximada da malha refinada

Inicialmente, somente as células próximas aos emissores são refinadas. À medida que a nuvem de poluente se aproxima da fronteira do domínio refinado, novas células do domínio da hidrodinâmica são refinadas aumentando a região onde é simulado o transporte de substâncias.

Ao refinar uma célula da hidrodinâmica, os valores de velocidade e nível em cada célula do domínio refinado necessários para o cálculo do transporte de substâncias são obtidos a partir dos dados da hidrodinâmica por algum esquema de interpolação. Após atualizados os valores da velocidade e nível, é gerado o sistema de equações para a resolução da concentração. O sistema é resolvido e os valores da concentração resultantes são atualizados nas células. Após atualizados, é verificado se o nível de concentração próximo a alguma das quatro fronteiras (considerando que o domínio do transporte é retangular) atingiu um valor pré-determinado. Se tal ocorreu, o domínio refinado é aumentado próximo a essa fronteira, sendo alocadas novas células próximas àquela fronteira. Como foi dito no início dessa seção, a hidrodinâmica e o transporte de substâncias são resolvidos em fases distintas. A seqüência de ciclos de hidrodinâmica e transporte é função do passo de tempo desejado, que por sua vez é função da resolução espacial utilizada.

O objetivo principal deste trabalho é a definição e implementação de esquemas de particionamento e balanceamento para problemas multi-fase como o HIDRA. Três métodos de balanceamento dinâmico de carga foram implementados e avaliados. No primeiro deles, o particionamento é baseado exclusivamente na malha da hidrodinâmica, e a malha de transporte é particionada de acordo com o particionamento da hidrodinâmica. A malha do transporte permanece acoplada sobre a malha da hidrodinâmica em cada subdomínio, ou seja, o processador que mantiver um subdomínio da malha da hidrodinâmica manterá também a malha refinada do transporte correspondente àquela região.

A segunda abordagem avaliada foi o particionamento independente das malhas da hidrodinâmica e do transporte e a migração das células da hidrodinâmica necessárias ao transporte no início de cada ciclo do transporte.

Na terceira abordagem avaliada, é feito inicialmente o particionamento da malha

do transporte e das células da hidrodinâmica onde ocorre transporte. Num segundo momento é feito o particionamento das células da hidrodinâmica onde não ocorre transporte. Uma descrição detalhada das três abordagens avaliadas bem como dos resultados obtidos é apresentada no capítulo 6.

2.5 Particionamento de Domínio em *Clusters* Heterogêneos

Quando a aplicação a ser particionada for executada em um *cluster* heterogêneo quanto à capacidade de processamento dos nodos, o particionamento do grafo da mesma deve levar em conta a capacidade de processamento dos nodos e a forma de conexão entre eles. Associado ao problema do particionamento está o problema de mapeamento do mesmo ao grafo da arquitetura. O problema de mapeamento em *clusters* heterogêneos foi abordado em (CARVALHO, 2002).

Quando a heterogeneidade do *cluster* se restringe à capacidade dos nodos, como pode ocorrer, por exemplo, quando um *cluster* sofre acréscimo de nodos, ou em um *cluster* montado a partir de máquinas de diferentes características, o problema da distribuição inicial da carga se reduz ao particionamento do grafo da aplicação em subgrafos proporcionais à capacidade de processamento dos nodos. Entre os pacotes de particionamento estudados o pacote JOSTLE (WALSHAW, 2000) permite o particionamento em subgrafos de tamanhos distintos, acrescentando ao vetor de pesos do grafo uma carga fictícia, de forma que após o particionamento alguns subgrafos terão uma carga real menor, sendo atribuídos esses subgrafos aos nodos de menor capacidade de processamento.,

Para a utilização do modelo em *clusters* heterogêneos quanto à capacidade de processamento mas totalmente conectados foi implementada uma versão do particionador por RCB que recebe como entrada um vetor com os tamanhos relativos dos subdomínios a serem gerados e utiliza essa informação para efetuar o particionamento. Esse vetor é gerado antes do particionamento a partir de um arquivo com as capacidades de processamento relativas dos nodos onde a aplicação será executada. Essa versão do particionador, a cada passo do particionamento, soma as capacidades de todos os nodos que pertencem ao domínio e as capacidades dos nodos que ficarão em cada subdomínio, fazendo o particionamento de forma proporcional. Essa versão do particionador não foi incorporada ao modelo, uma vez que os *clusters* utilizados no desenvolvimento e durante a fase de testes do modelo são homogêneos em relação à capacidade de processamento dos nodos.

3 BALANCEAMENTO DINÂMICO DE CARGA

Problemas de balanceamento de carga podem ser classificados de acordo com a granularidade da aplicação, que deve ser levada em conta ao escolher-se um mecanismo de balanceamento. Aplicações de grão fino normalmente consistem de um número grande de processos pequenos ou itens de dados independentes que podem ser facilmente transferidos entre processadores. Aplicações de grão grosso (*coarse-grain*) consistem de poucos processos mais pesados e normalmente são mais difíceis de transferir. Diekmann e Monien (DIEKMANN; MONIEN; PREIS, 1997) dividem os problemas de balanceamento dinâmico em 4 classes, baseando-se na estrutura de comunicação da aplicação e na possibilidade de migrar processos. As 4 classes de problemas identificados por eles são:

- problemas de mapeamento dinâmico, nos quais processos são gerados dinamicamente e devem ser mapeados nos processadores. Normalmente não possuem dependência entre eles mas não podem ser migrados. Usualmente surgem em aplicações cliente-servidor;

- problemas de aninhamento dinâmico (*dynamic embedding problem*), quando ocorre a criação dinâmica de processos que não podem ser migrados e entre os quais ocorrem fortes dependências. Surgem, por exemplo, em problemas de busca em espaço de estados e uma abordagem utilizada para balanceamento dos mesmos é a colocação dos processos nos nodos próximos ao nodo a partir de onde o processo foi criado;

- problemas de balanceamento dinâmico de carga, são aplicações onde tarefas sem dependência entre si e que podem ser migradas são geradas dinamicamente. Existe um número grande de abordagens para essa classe de problemas. Casavant e Kuhl (CASAVANT; KUHL, 1988) apresentam uma taxonomia para algoritmos de escalonamento baseado em critérios como localidade e optimalidade;

- problemas de re-aninhamento dinâmico (*Dynamic re-embedding problems*) são problemas em que o grafo da aplicação é dinâmico, ou seja, arestas e vértices são gerados ou removidos durante a execução. Devido à dependência entre os subdomínios, a abordagem utilizada nos problemas de balanceamento dinâmico de carga, de atribuir as tarefas aos processadores baseando-se apenas na carga, sem considerar a conexão entre os mesmos, não é utilizada, pois rapidamente aumentaria a quantidade de comunicação entre todos os processadores. Nesse capítulo serão discutidas soluções para essa categoria de problemas, na qual se enquadra o problema estudado nesse trabalho.

3.1 Reparticionamento de domínio

Em certos modelos, como resultado da ocorrência de fenômenos físicos como altos gradientes que exijam um refinamento da malha em partes do domínio, a malha é alterada durante a execução. Neste caso, para ser mantida a boa distribuição da carga entre os processadores a malha deve ser reparticionada. A nova partição deve atender as seguintes condições (SCHLOEGEL; KARYPIS; KUMAR, 2000): manter a distribuição de carga bem balanceada, minimizar a comunicação e minimizar a quantidade de movimentação de dados para realizar a nova partição.

Existem duas abordagens principais para o reparticionamento. Uma delas é calcular uma partição inteiramente nova, a partir do zero. Esta abordagem pode resultar em baixo desempenho além de eventualmente necessitar de uma quantidade grande de movimentação de dados, caso a nova partição seja muito diferente da anterior. Esquemas conhecidos como *scratch-remap* (SCHLOEGEL; KARYPIS; KUMAR, 1999) calculam uma nova partição e mapeiam a nova partição para a partição existente. Estes algoritmos funcionam bem para grafos com um desbalanceamento alto concentrado em algumas regiões do domínio. Em domínios com um desbalanceamento baixo ou bem distribuído em todo o domínio, esta abordagem pode resultar em uma transferência excessiva de células, se comparado aos algoritmos de difusão multinível, descritos na seção 3.2.

A outra abordagem é o uso de algoritmos de difusão. Estes algoritmos, de forma geral, garantem uma partição de boa qualidade, com pouca comunicação entre os subdomínios e com pouca movimentação de dados para a realização da nova partição (SCHLOEGEL; KARYPIS; KUMAR, 2000). Em domínios com um desbalanceamento alto concentrado em determinadas regiões, algoritmos difusivos podem resultar em uma migração excessivamente alta de células, devido à propagação da difusão em grandes áreas.

O uso de algoritmos difusivos para balanceamento de carga foi apresentado pela primeira vez por Cybenko (CYBENKO, 1989) e Boillat (BOILLAT, 1990). Neles, a cada passo de tempo, cada processador avalia sua carga e a compara com a dos seus vizinhos. Se a diferença de carga estiver acima de um determinado valor, parte da carga é transferida de um processador para outro, tal como ocorre na difusão de calor. Migração de carga por difusão é descrita em mais detalhes na seção 3.2.

Balanceamento dinâmico pode ser obtido de diversas formas, dependendo da aplicação e do ambiente considerados. Ele pode ser tratado pelo sistema operacional, através da migração de processos ou *threads* ou pela aplicação. Watts (WATTS, 1995) divide o processo de balanceamento de carga em 4 fases:

1. Avaliação de carga - É utilizada alguma medida da carga dos processadores para verificar se a carga está balanceada;

2. Avaliação da relação custo-benefício (*profitability*) de efetuar o balanceamento de carga - Após a medida da carga dos processadores, a presença de desbalanceamento pode ser detectada. Se o custo de manter a carga desbalanceada é maior que o custo de balanceá-la, o processo de balanceamento é iniciado;

3. Cálculo da carga a ser transferida - Baseado nas cargas medidas na primeira fase, a quantidade ideal de carga a ser transferida para balanceá-la é calculada;

4. Migração da carga - A carga é transferida entre os processadores.

A forma como esse processo é implementado depende da aplicação e ambiente utilizado. Na área de aplicação abordada neste trabalho, onde o paralelismo é obtido por particionamento de domínio, as duas últimas fases são representadas pelo cálculo de uma nova partição e migração dos dados para o novo particionamento. Nas próximas seções serão discutidos rapidamente cada uma das quatro etapas.

3.1.1 Avaliação de carga

A avaliação da carga de cada processador é necessária para detectar o grau de desbalanceamento da aplicação, bem como para identificar entre quais processadores deve ser feita a migração de carga e qual a carga que deve ser transferida. A avaliação da carga pode ser feita analiticamente, empiricamente ou por uma combinação de ambos. Na avaliação analítica, a estimativa do custo de processamento no nodo é feita a partir de um modelo matemático da aplicação. Essa abordagem tem a vantagem de poder fornecer uma avaliação a priori da distribuição da carga, podendo ser útil para definir a distribuição inicial de carga. A desvantagem dessa abordagem é a dificuldade de se obter um modelo acurado da aplicação. Fatores dependentes do sistema podem alterar significativamente o tempo de execução da aplicação, tornando a avaliação analítica inaceitavelmente imprecisa.

Entre os fatores que podem afetar o tempo de execução pode-se citar o uso da memória *cache* e anomalias na paginação. Quando uma aplicação é dividida entre diversos processadores, a quantidade de dados utilizada por cada é reduzida proporcionalmente ao número de processadores. Isto faz com que, como o processador trabalha com menos dados, a taxa de acertos na memória *cache* aumente, reduzindo o tempo médio de acesso à memória (média ponderada entre o tempo de acesso da memória *cache* e da memória principal) e, conseqüentemente, reduzindo o tempo necessário para processar o subdomínio do processador. Este fenômeno, que também ocorre a nível de paginação, é difícil de modelar analiticamente e explica em diversas situações, a obtenção de *speedups* superlineares ao paralelizar aplicações (por exemplo, ver (CANAL, 2001)). A importância da memória *cache* no desempenho de aplicações é mostrado por Zagha (ZAGHA et al., 1996), que obteve um ganho de desempenho de 33% em uma simulação usando o método ADI tridimensional, simplesmente trocando a ordem de alguns laços e alterando a direção da varredura da matriz, o que não alterou o número de operações mas aumentou significativamente a taxa de acertos na memória *cache*. Como exemplo de modelo matemático de um método numérico, Chan (CHAN; SHAO, 1994) apresenta um modelo matemático do custo computacional de um gradiente conjugado pré-condicionado e o utiliza para encontrar o tamanho ótimo de grade em um método multinível.

A avaliação empírica, baseada na medição direta da carga, permite obter uma estimativa real da carga do processador. Normalmente os sistemas fornecem mecanismos para temporização com precisão de milisegundos a microsegundos. Esses mecanismos podem ser utilizados para obter medidas precisas do tempo de execução, tempo ocioso e tempo de comunicação. Pode ser utilizado um mecanismo em que cada processador avalie o percentual de tempo utilizado em processamento e, quando este percentual estiver abaixo de um determinado valor, solicite aos outros processadores uma análise da necessidade de iniciar uma redistribuição da carga. A avaliação direta de carga é útil em situações em que a carga passada do processador pode ser usado para estimar sua carga futura.

A maior parte dos processadores atuais possui contadores internos de eventos

como número de falhas da *cache* e número de operações de ponto flutuante executadas (ZAGHA et al., 1996). Estes contadores, que nas versões mais recentes dos processadores passaram a ser disponibilizados para aplicações (o primeiro processador da Intel a disponibilizar estes contadores foi o Pentium Pro, em 1995, através da instrução RDPMC - *Read Performance Monitoring Counters* (WEBBER, 2000)), podem ser utilizados para avaliação de desempenho de aplicações bem como para avaliação de carga.

Na avaliação híbrida, medidas empíricas são utilizadas para obter as constantes utilizadas nos modelos analíticos, de modo a melhorar a acurácia dos mesmos.

3.1.2 Detecção de desbalanceamento

Uma vez feita a avaliação de carga dos processadores, é necessário avaliar em que ponto vale a pena iniciar o processo de redistribuição de carga. A redistribuição de carga representa carga adicional para a aplicação e não deve ser feita com freqüência maior que a necessária. Por outro lado, se a redistribuição de carga for feita com uma freqüência muito baixa, o desbalanceamento pode afetar o desempenho da aplicação. Por essa razão é importante achar um ponto ótimo a partir do qual deve ser disparado o processo de redistribuição. Existem numerosos esquemas para aplicações baseadas em paralelismo de tarefas (*task-parallelism*)(CAMPOS; SCHERSON, 2000)(WILLEBEEK-LEMAIR; REEVES, 1993)(SHU; WU, 1995) mas os resultados para aplicações baseadas em paralelismo de dados (*data-parallelism*) ainda são poucos (CERMELE; COLAJANNI; TUCCI, 1997).

O processo de decidir o momento de efetuar o rebalanceamento pode ser feito em duas fases: detectando desbalanceamentos e determinando se o ganho da nova distribuição compensa o custo da redistribuição da carga.

Para a detecção, cada processador pode calcular seu percentual de trabalho útil e comparar com a média dos outros processadores. Se um número significativo de processadores estiver com uma carga muito acima ou abaixo da média, é iniciado o processo de redistribuição. O tempo necessário para redistribuir a carga pode ser estimado do histórico das redistribuições anteriores ou calculado de forma analítica. A redução no tempo de execução pode ser aproximada supondo que se chegará a uma eficiência de 100% ou utilizando um histórico do ganho obtido com as redistribuições anteriores. A detecção de desbalanceamento e comparação com os outros processadores pode ser feita de forma síncrona, em aplicações que possuem pontos de sincronismo, ou de forma assíncrona, a qualquer momento em que um processador detecte o desbalanceamento. Aravinthan (ARAVINTHAN et al., 1999) propõe uma heurística para a decisão de redistribuição de carga baseada em uma função t_{cost}, que modela o tempo para redistribuição e o tempo de execução da aplicação em relação à taxa de aumento do desbalanceamento. Biswas (BISWAS; OLIKER; SOHN, 1996) calcula o ganho obtido com a redistribuição como T_{iter} * N_{adapt} * (W_{maxant}-$W_{maxnova}$), onde T_{iter} é o tempo de cada iteração de cálculo, N_{adapt} é o número estimado de ciclos antes de ser necessária nova redistribuição dos dados e W_{max} é a carga do nodo de maior carga do sistema. O custo da redistribuição é calculado a partir dos parâmetros de latência e velocidade da rede e quantidade de dados a ser transferida. Se o ganho obtido com a redistribuição supera o seu custo, a mesma é iniciada.

3.1.3 Cálculo da carga a ser transferida

Após determinar que é vantajoso reequilibrar a carga, a quantidade de carga que deve ser transferida de um processador para outro deve ser calculada. Para isso pode ser utilizado um modelo analítico da aplicação ou alguma medida como o tempo utilizado para processar sua carga em comparação com os processos vizinhos.

3.1.4 Transferência de carga

Esta fase é responsável pela movimentação dos dados quando mapeados para outro processador. Dois tipos de *overhead* podem ocorrer nessa fase: comunicação e computação (BISWAS; OLIKER; SOHN, 1996). O *overhead* de comunicação inclui o custo de empacotar e desempacotar dados, tempo de latência e transmissão das mensagens. O custo de computação é o tempo necessário para selecionar as células a serem transferidas e reorganizar as estruturas de dados de maneira consistente. Na aplicação abordada neste trabalho, a carga transferida corresponde a conjuntos de células do domínio com todas as variáveis associadas. Isto implica em alocação e desalocação das mesmas e atualização das estruturas que descrevem os subdomínios, podendo essa reorganização ter um custo significativo.

3.2 Migração de carga por difusão

Nesta seção será apresentada uma visão geral dos métodos de difusão e sua aplicação para balanceamento de carga sobre grafos de processadores. Inicialmente serão apresentadas algumas definições e após serão apresentados os esquemas de difusão de primeira e segunda ordem.

Um conjunto de P processadores ligados por uma rede de interconexão pode ser representado por um grafo G(V,E), com $|V|$ vértices e $|E|$ arestas onde os vértices representam os processadores e as arestas representam a conexão entre eles. Vértices são numerados de 0 a *P-1* e as arestas são numeradas de 1 a $|E|$. Considera-se que o grafo é conexo (caso contrário, a carga não pode migrar entre as componentes conexas do mesmo e o problema só pode ser resolvido dentro de cada componente conexa). Associado a cada vértice i está um valor escalar w_i, representando a carga do processador. A carga média $\overline{w}$ de todos os processadores é dada por $\overline{w} = \frac{\sum_{i=0}^{P-1} w_i}{P}$. Alternativamente, em um modelo como o apresentado neste trabalho, os vértices podem representar os subdomínios, com o peso w_i de cada subdomínio representando o número de células do subdomínio e as arestas representando a relação de vizinhança entre os subdomínios.

O problema de balanceamento dinâmico de carga é o de encontrar a quantidade de carga δ_{ij} que deve ser transferida entre cada par de processadores i e j de forma que após a transferência a carga esteja em equilíbrio. Esse problema pode ser resolvido diretamente através da solução do sistema Ax=b, onde $x = (\delta_1, \delta_2, \ldots, \delta_{|E|})^T$ é o vetor com a carga que deve ser transferida através de cada aresta (i, j), b é o vetor que contém, para cada vértice do grafo, a diferença entre sua carga e a carga média esperada $\overline{w}$ dado por

$$b = (w_1 - \overline{w}, w_2 - \overline{w}, \ldots, w_{|V|} - \overline{w})^T$$

e A é a matriz de incidências do grafo, de ordem $|V|$ x $|E|$, definida por

$$(A)_{i,j} = \begin{cases} 1, & \text{se vértice i é o destino da aresta j} \\ -1, & \text{se vértice i é origem da aresta j} \\ 0, & \text{se vértice i não é nem o destino e nem a origem da aresta j} \end{cases}$$

Como normalmente o número de arestas do grafo é maior que o número de vértices, o número de equações do sistema é menor que o número de incógnitas e o sistema pode apresentar infinitas soluções. Nesse caso, pode-se buscar, entre as soluções existentes, aquela que atende a algum outro critério, como a minimização da norma euclidiana da quantidade de carga a ser transferida. O exemplo a seguir, criado sobre o modelo do Guaíba, ilustra essa situação. A figura 3.1 mostra o particionamento do Guaíba entre 8 subdomínios, um subdomínio para cada processo, e a figura 3.2 mostra o grafo representando uma possível distribuição de carga entre eles. Os valores ao lado de cada vértice representam a carga de cada processador. Os valores são fictícios, mas podem representar, por exemplo, o número de células em cada subdomínio.

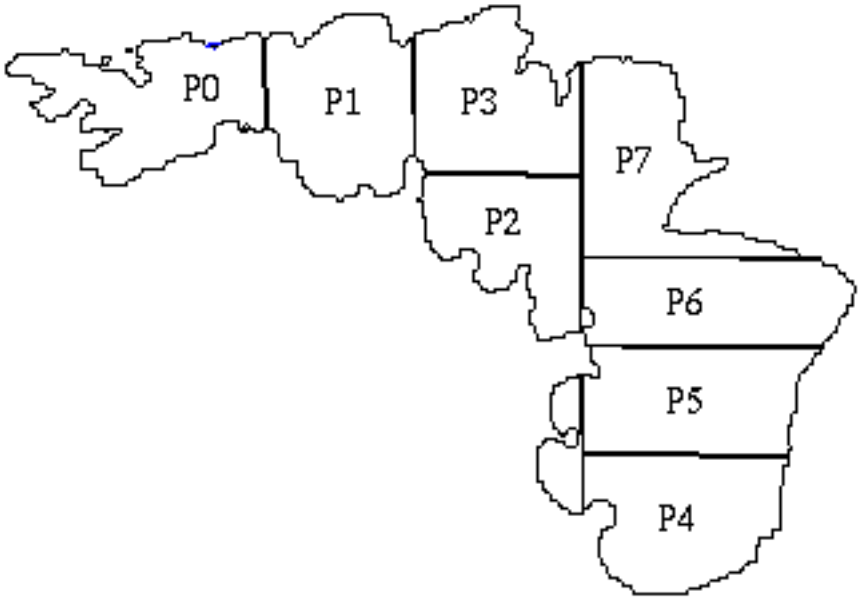


Figura 3.1: Domínio do Guaíba particionado em 8 subdomínios

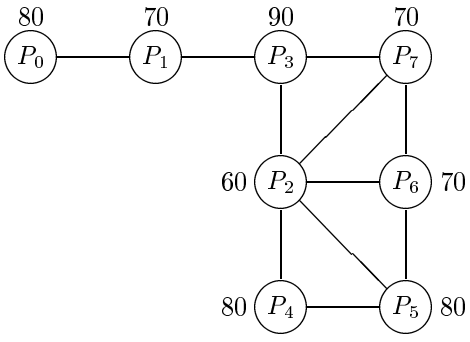


Figura 3.2: Grafo associado ao Guaíba particionado em 8 subdomínios

No exemplo mostrado na figura 3.2a soma das cargas de todos os processos é 600, que dividido pelos 8 processos resulta em uma carga média de 75 por processo. Existem diversos esquemas de transferência de carga para chegar a essa carga média e a figura 3.3 ilustra dois exemplos possíveis de transferência de carga para chegar a essa distribuição ideal. Numa situação dessas pode-se buscar, entre as soluções existentes, aquela que atende a algum outro critério, como a minimização da norma euclidiana da quantidade de carga a ser transferida.

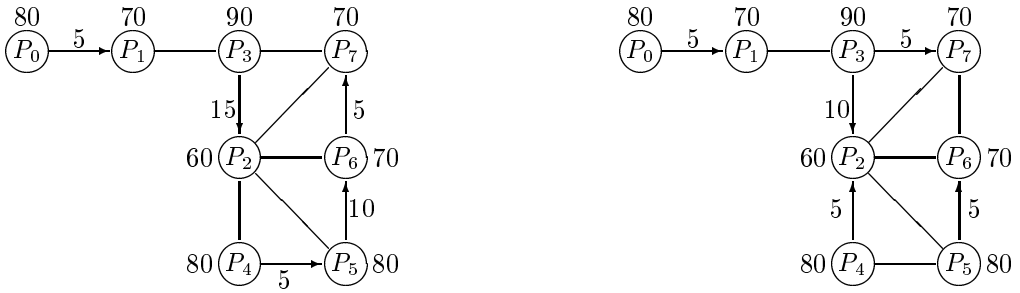


Figura 3.3: Dois diferentes esquemas de transferência de carga

Hu e Blake (HU; BLAKE, 1998)(HU; BLAKE, 1999) mostram que a minimização da norma euclidiana da quantidade de carga transferida pode ser obtida pela minimização da função $\frac{1}{2}x^T x$ (HU; BLAKE, 1998) sujeita à restrição $Ax = b$ e a solução da mesma é dada por $x = A^T\lambda$ onde λ é o vetor de multiplicadores de Lagrange do grafo. Substituindo a segunda expressão na primeira obtem-se a expressão $L\lambda = b$ onde $L = AA^T$ é a matriz Laplaciana do grafo definida por

$$(L)_{ij} = \begin{cases} -1 & \text{se } i \neq j, \text{ e existe a aresta } (i,j) \\ grau(i) & \text{se } i = j \\ 0 & \text{em caso contrário} \end{cases}$$

Assim, o problema de encontrar o esquema de transferência de carga que minimize a norma euclidiana da quantidade de carga transferida pode ser resolvido pela solução da equação $L\lambda = b$. A resolução desse sistema pode ser feita por um único processo que receba informação sobre a conectividade e carga de todos os outros e que envie a cada um a quantidade de carga a ser trocada com cada vizinho, ou as informações sobre carga e conectividade de todos os processos podem ser trocadas entre eles de modo que todos os processos possam resolver o sistema. Ambas as soluções necessitam de comunicação global para a troca de informações, o que pode prejudicar o desempenho e a escalabilidade da solução. Esse uso de informações globais não ocorre nos esquemas difusivos, que serão apresentados agora.

Nos esquemas difusivos de primeira ordem (XU; LAU, 1994) (FOS - *First Order Scheme*), uma fração α_{ij} da diferença de carga $w_i - w_j$ é movida ao longo da aresta (i,j). Considerando w^t o vetor com a distribuição de carga entre os vértices após o passo de tempo t, w_0 é o vetor com a distribuição inicial e a carga em um passo t+1 pode ser modelada como $w^{t+1} = D(\alpha)w^t$, onde $D(\alpha) = (d_{ij})$ é uma matriz de difusão, real, não negativa, simétrica e duplamente estocástica (a soma de cada linha ou coluna é igual a 1) definida por

$$d_{ij} = \begin{cases} \alpha & \text{se } i \neq j, \text{ e existe a aresta } (i,j) \\ 1 - \delta(i)\alpha & \text{se } i = j \\ 0 & \text{em caso contrário} \end{cases}$$

onde δ é o grau do vértice i, ou seja, o número de processadores conectados diretamente ao processador i. Por exemplo, para uma malha uni-dimensional com 4 nodos, a matriz de difusão é dada por:

$$D(\alpha) = \begin{Vmatrix} 1-\alpha & \alpha & & \\ \alpha & 1-2\alpha & \alpha & \\ & \alpha & 1-2\alpha & \alpha \\ & & \alpha & 1-\alpha \end{Vmatrix}$$

A partir desse modelo, a quantidade de carga l_{ij}^{t+1} a ser movida de um nodo i para um nodo j no passo t+1 é dada por $l_{ij}^{t+1} = \alpha_{ij}(w_i^t - w_j^t)$

Um aspecto importante dos métodos de difusão é a sua convergência, ou seja, se será atingida uma situação onda a carga esteja bem distribuída e qual é o número de passos necessários para atingir essa situação. Diekmann et al. (DIEKMANN; MONIEN; PREIS, 1997) mostram que se o segundo maior autovalor em valor absoluto da matriz de difusão é menor que 1, o esquema converge, caso contrário, o esquema diverge. O modelo de difusão apresentado aqui considera um único valor de α para toda a matriz de difusão. Para algumas arquiteturas, como o Hipercubo, a matriz de difusão ótima (no sentido de melhor convergência) utiliza um α constante, o mesmo não valendo para outras, como cilindros e grades (DIEKMANN; MONIEN; PREIS, 1997). Para o caso genérico, Boillat (BOILLAT, 1990) sugere

$$\alpha_{ij} = \frac{1}{max\{grau(i), grau(j)\}+1}, i, j \in V$$

Outro fator que influencia a velocidade de convergência é o grau de conectividade do grafo, sendo que quanto maior o número de arestas do grafo, mais rápida é a convergência. Boillat (BOILLAT, 1990) provou que a convergência mais lenta ocorre quando os processadores estão dispostos em linha. Neste caso o número de iterações necessárias para atingir uma dada tolerância é $O(P^2)$, onde P é o número de processadores.

Hu e Blake (HU; BLAKE, 1999) desenvolveram um algoritmo de difusão que apresenta melhor convergência, mas para o qual é necessário o cálculo do maior e menor autovalores do Laplaciano do grafo de carga. O algoritmo utiliza comunicação global no início de cada ciclo de difusão e, após, utiliza apenas comunicação local, exceto se for necessário um critério de convergência global. Outra variação dos esquemas difusivos são os esquemas difusivos de segunda ordem, nos quais a quantidade de carga enviada no passo anterior é levada em conta para evitar oscilações, ou seja, situações onde carga fica sendo transferida entre dois ou mais nodos sem chegar a uma situação de estabilidade, que podem ocorrer em esquemas de primeira ordem (GHOSH; MUTHUKRISHNAN; SCHULTZ, 1996). Nos esquemas de segunda ordem (SOS - *Second Order Scheme*), cada iteração é modelada da seguinte forma:

$$w^{k+1} = \begin{cases} Mw^k & \text{se } k = 0 \\ \beta Mw^k + (1-\beta)Mw^{k-1} & \text{se } k > 0 \end{cases}$$

sendo β um coeficiente independente do número da iteração. Em relação à convergência, esquemas de segunda ordem convergem se β está no intervalo $(-\infty, 0)$ ou $(2, +\infty)$. No intervalo (0,1), o esquema de segunda ordem converge mais lentamente

que o esquema de primeira ordem. No intervalo (1,2) existe um valor de β para o qual esquemas de segunda ordem convergem mais rapidamente que esquemas de primeira ordem (DIEKMANN; MONIEN; PREIS, 1997).

Schloegel, Karypis e Kumar utilizaram, na biblioteca Métis, uma abordagem multinível para a difusão. Na abordagem utilizada, o grafo é reduzido, através de contrações de vértices, a um grafo menor. Esse grafo é, então, balanceado através de difusão e refinado utilizando um k-refinamento para melhorar a qualidade. O grafo assim obtido é refinado para o próximo nível e o processo se repete até voltar ao grafo original. Em (SCHLOEGEL; KARYPIS; KUMAR, 1998) Schloegel e Karypis compararam métodos difusivos com métodos de remapeamento, utilizando o ParMétis, uma versão paralela da biblioteca Métis de particionamento de grafos, onde mostraram que algoritmos de difusão multinível resultam em menor migração de dados para desbalanceamentos pequenos ou não localizados. Os algoritmos de remapeamento, por outro lado, em problemas de desbalanceamentos localizados, resultam numa partição de melhor qualidade, mantendo o mesmo *overhead* de migração de vértices que os esquemas difusivos. Quanto ao tempo de execução, ambas as abordagens mostraram ser similares.

A escolha da abordagem a ser utilizada, como é de se esperar, depende das características o problema. Em problemas em que as alterações na malha são moderadas, como é o caso do problema abordado nesse trabalho, a difusão tem um custo menor que o cálculo de todo o particionamento.

No *cluster* utilizado neste trabalho, os processadores são ligados entre si por um *switch*, o que possibilita que todos se comuniquem com todos (mas não simultaneamente), e cada processo detém um ou mais subdomínios. Na modelagem do problema, cada processo (e os subdomínios por ele tratados) está associado a um vértice e a carga do vértice representa o número de células dos seus subdomínios. As arestas do grafo representam o número de células de fronteira entre os subdomínios presentes nos dois processadores adjacentes àquela aresta. Assim, dois subdomínios que não têm uma fronteira comum não possuem no grafo uma aresta associada a essa fronteira. Esse modelo foi adotado porque a transferência de células entre subdomínios não adjacentes acabaria fragmentando o domínio em diversos subdomínios pequenos pertencentes ao mesmo processador acarretando maior custo computacional total (devido à maior área de sobreposição entre os subdomínios) e possivelmente piorando a convergência dos métodos de solução baseados em decomposição de domínio.

No particionamento por faixas que foi utilizado no modelo de transporte de substâncias desse trabalho, o grafo da aplicação é um grafo linear, onde cada vértice n é adjacente aos vértices $n-1$ e $n+1$, exceto o vértice 0 que é adjacente somente ao vértice 1, e o vértice $P-1$, correspondente ao último subdomínio, que é adjacente apenas ao vértice $P-2$. Para esse grafo, ao contrário do grafo do exemplo anterior, há apenas uma solução para o problema de transferência de carga e pode ser obtida diretamente por cada processador se o mesmo possuir informação sobre a quantidade de carga de cada um dos outros processadores. Esse esquema foi implementado e os resultados obtidos serão analisados posteriormente.

4 MODELO DE HIDRODINÂMICA E TRANSPORTE DE SUBSTÂNCIAS NO RIO GUAÍBA

O GMCPAD e o GPPD do Instituto de Informática da UFRGS vêm trabalhando no estudo e desenvolvimento de modelos paralelos de hidrodinâmica e de transporte de substâncias em *clusters* de máquinas multiprocessadas. O desenvolvimento deste tipo de modelo envolve diversos aspectos como a implementação paralela de métodos numéricos, métodos de decomposição de domínio e balanceamento de carga. Neste capítulo serão descritos brevemente os modelos matemáticos utilizados no modelo de circulação e transporte de substâncias desenvolvidos pelo grupo e sobre o qual foram implementados e testados os mecanismos de particionamento e de balanceamento de carga que são objeto deste trabalho. Um detalhamento maior pode ser encontrado em (RIZZI, 2002).

4.1 Modelo matemático e condições de contorno

Modelos de simulação de fenômenos físicos como o deste trabalho são baseados em modelos matemáticos que descrevem os fenômenos físicos envolvidos. Muitos desses modelos são equações diferenciais parciais que normalmente não tem soluções analíticas, sendo aproximadas através de modelos discretos. Neste trabalho desenvolveram-se algoritmos e soluções para o problema de particionamento e de balanceamento dinâmico de carga para um modelo de circulação e transporte de substâncias 2D e 3D desenvolvido pelo grupo. O modelo desenvolvido neste trabalho é baseado nas Equações de Shallow Water (RIZZI, 2002), que podem ser escritas como:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} + w\frac{\partial u}{\partial z} = -g\frac{\partial \eta}{\partial x} + \nu_h(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}) + \frac{\partial}{\partial z}(\nu_v\frac{\partial u}{\partial z}) + \Phi v$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} + w\frac{\partial v}{\partial z} = -g\frac{\partial \eta}{\partial y} + \nu_h(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}) + \frac{\partial}{\partial z}(\nu_v\frac{\partial v}{\partial z}) - \Phi u$$

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} + \frac{\partial w}{\partial z} = 0$$

onde

- u=u(x,y,z,t) e v=v(x,y,x,t) são as velocidades horizontais nas direções X e Y, e w=w(x,y,z,t) é a velocidade vertical na direção Z;
- $\eta = \eta(x, y, t)$ é a altura da água acima de um nível de referência;

- $\Phi = 2\omega sen\phi$ é a força de Coriolis, onde ω é a velocidade angular da Terra e ϕ é a latitude;

- $\nu_h = \nu_h(x, y, z, t)$ é o coeficiente de viscosidade cinemática turbulenta horizontal;

- $\nu_v = \nu_v(x, y, z, t)$ é o coeficiente de viscosidade cinemática turbulenta vertical;

- g é a aceleração gravitacional, e t é o tempo.

A equação para a integração entre as camadas é obtida integrando-se a equação da continuidade sobre a coluna de água e usando-se a condição de contorno cinemática na superfície livre e é dada por

$$\frac{\partial \eta}{\partial t} + \frac{\partial}{\partial x}(\int_{-h}^{\eta} u dz) + \frac{\partial}{\partial y}(\int_{-h}^{\eta} v dz) = 0$$

onde $h = h(x, y)$ é a profundidade da água medida a partir de um dado nível.

Sendo um problema de valor inicial e de contorno, é necessário especificar essas condições. Nos contornos fechados a condição imposta foi que velocidades e níveis de água são ambos nulos. Nas fronteiras abertas foram especificados dois tipos de condições de contorno. O tipo nível e o tipo fluxo (velocidade ou vazão).

A equação de transporte escalar de substâncias 3D (ETM3D) é o modelo matemático para o transporte (advecção e difusão) de substâncias dispersas em água. É uma EDP hiperbólica não linear e é escrita como:

$$\begin{aligned} \frac{\partial(S)}{\partial t} + \frac{\partial(uS)}{\partial x} + \frac{\partial(vS)}{\partial y} + \frac{\partial(wS)}{\partial z} &= \frac{\partial}{\partial x}(\varepsilon_h \frac{\partial S}{\partial x}) + \frac{\partial}{\partial y}(\varepsilon_h \frac{\partial S}{\partial y}) \\ &\quad + \frac{\partial}{\partial z}(\varepsilon_v \frac{\partial S}{\partial z}) + FS + K \end{aligned}$$

onde S é a concentração; ε_h=ε_h(x,y,z,t) é o coeficiente de difusão turbulenta horizontal; ε_v=ε_v(x,y,z,t) é o coeficiente de difusão turbulenta vertical; F é o termo de fonte e K=2,3/T_{90}, em que T_{90} é o tempo para remover 90 % dos organismos, é a matriz de decaimento. Esses coeficientes, bem como os das equações da hidrodinâmica são fornecidos por um modelo de parametrização de turbulência.

A solução da ETM também requer a especificação de condições de contorno dos tipos de Dirichlet e de Neumann. A CC de Dirichlet é definida especificando uma concentração nas fronteiras onde ocorre entrada de água para um período específico de tempo, de modo a agir como uma fonte, fornecendo massa para o domínio, ou como sumidouro, retirando a massa do domínio. Neste caso, a condição de contorno é dada por S=S(x,y,z,t), em que $(x, y, z, t) \in \partial\Omega x[0, T]$, onde $\partial\Omega$ denota a fronteira do domínio e S(x,y,z,t) é uma concentração nessa fronteira. A CC tipo Neumann ocorre quando o gradiente de concentração é normal à fronteira, de modo que um fluxo dispersivo é aí especificado. Essa CC é expressa como $\frac{\partial S}{\partial n}$, em que n=n(x,y,z,t) é o vetor unitário normal no ponto $(x, y, z, t) \in \partial\Omega$. Neste trabalho tomou-se o gradiente nulo, $\nabla S_{\partial\Omega} = 0$, na fronteira onde ocorre saída de água, indicando que a taxa de variação espacial é nula, de modo que as substâncias possam sair do domínio sem acumular-se nessa fronteira.

4.1.1 Discretização: esquema numérico semi-implícito e malha alternada

As equações que descrevem os fenômenos físicos são contínuas e não tem solução analítica. Para a solução das mesmas é necessário que sejam discretizadas, bem como o domínio do problema. Nesse processo de discretização os termos das equações são aproximados e os sistemas de equações resultantes desse processo de discretização são resolvidos utilizando métodos numéricos. Existem diversas classes de métodos para fazer estas aproximações, entre os quais o método de diferenças finitas, o método dos elementos finitos e o método de volumes finitos. Neste trabalho foram utilizadas diferenças finitas por ser um método suficientemente adequado para os propósitos deste trabalho, além de possuir características que permitem uma boa exploração do paralelismo das máquinas multiprocessadas. O esquema numérico foi definido sobre uma malha alternada (*staggered*) Arakawa classe C (MESSINGER, 1998), mostrada na figura 4.1. O desacoplamento das variáveis que não necessitam ser definidas no mesmo ponto resulta na redução do tempo de processamento, com a vantagem do erro de truncamento permanecer o mesmo (MESSINGER, 1998).

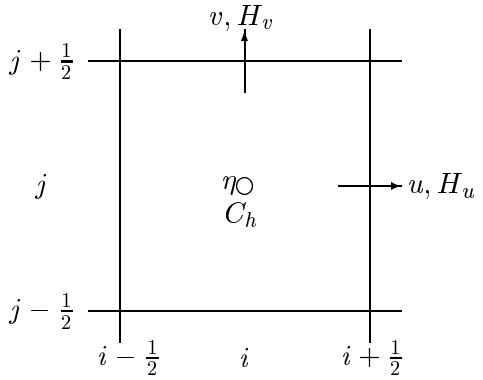


Figura 4.1: Malha regular alternada tipo Arakawa C.

A abordagem para construir o esquema numérico utilizado no primeiro modelo desenvolvido neste trabalho foi uma abordagem semi-implícita, na qual discretizam-se alguns termos das EDPs governantes de forma implícita e outros termos de forma explícita. Adotando-se essa abordagem para a discretização das ESWs, os sistemas de equações resultantes são lineares e algébricas (SELAs), cujas matrizes dos coeficientes são estruturadas e esparsas, contendo no máximo 5 elementos por linha. Essas matrizes podem ser resolvidas eficientemente por métodos iterativos do subespaço de Krylov. Além disso, usando-se o método ADI (*Alternating Direction Implicit*) pode-se obter matrizes tridiagonais que são resolvidas eficientemente pelo método de Thomas, uma variante do método de eliminação de Gauss.

Neste trabalho foram construídos esquemas semi-implícitos que resultam em matrizes simétricas definidas positivas (SDP), de tal forma que é possível empregar o método do gradiente conjugado. Para isso, seguiu-se a estratégia desenvolvida por Casulli (CASULLI, 1990). Ela consiste em desenvolver um esquema semi-implícito onde as incógnitas são as velocidades da água e os níveis. Então, substituindo-se os termos das velocidades e das equações do momento discretas na equação da continuidade discreta obtém-se SELAs cujas matrizes são SDP.

4.1.2 Modelos computacionais e ambiente de desenvolvimento

Durante esse trabalho, foram desenvolvidos diversos modelos computacionais bidimensionais (2-D) e tridimensionais (3-D) paralelos para a hidrodinâmica e para o transporte de substâncias, diferenciando-se entre eles pelo método de paralelização utilizado, pelo esquema de discretização, pelo *solver* ou por características que foram sendo incorporadas aos modelos como batimetria ou desalocação de regiões da malha onde a concentração ficava abaixo de um determinado valor. Foram feitos experimentos com modelos baseados na estratégia de decomposição de domínio usando o método aditivo de Schwarz e com modelos baseados na abordagem de decomposição dos dados. Os *solvers* implementados para os modelos foram o Gradiente Conjugado com e sem pre-condicionamento (SHEWCHUCK, 1994) para resolução de matrizes definidas localmente e globalmente, o GMRES (SAAD, 1996) e o método de Thomas (POVITSKY, 1998).

A decisão de implementar diversos modelos computacionais deve-se ao fato de averiguar quais das associações de esquema numérico, estratégia de paralelização e método de solução são as mais convenientes, tanto do ponto de vista computacional quanto do ponto de vista da qualidade da simulação oferecida pelo modelo. Os esquemas numéricos e *solvers* dos diversos modelos implementados são descritos em (RIZZI, 2002), não sendo apresentados aqui por não terem influência direta nos esquemas de balanceamento de carga aqui apresentados. Ao invés disso, serão apresentadas as estruturas utilizadas para descrição dos domínios e armazenamento das malhas, necessárias para a boa compreensão dos esquemas de balanceamento implementados.

4.1.3 Descrição das estruturas utilizadas para armazenar o domínio

Nessa seção serão apresentadas as estruturas utilizadas para descrição e armazenamento do domínio. Como será visto na seção 4.1.4, a descrição do domínio é lida de um arquivo e armazenada em estruturas a partir das quais é feito o particionamento do domínio e a distribuição das descrições dos subdomínios aos processos que os irão manter inicialmente. Uma vez que cada processo tenha a descrição de seu subdomínio, ele aloca a malha correspondente e inicializa as variáveis necessárias em cada célula da malha. A seguir são descritas as três principais estruturas envolvidas nesse processo, que são o arquivo de descrição de todo o domínio, as estruturas que armazenam a estrutura do domínio em memória e as malhas da hidrodinâmica e transporte. A descrição dessas estruturas nesse texto visa facilitar a compreensão dos algoritmos de particionamento e balanceamento.

- Arquivo de descrição do domínio: O domínio é descrito, no arquivo de entrada, como uma série de conjuntos de células contíguas na direção X, seguida por uma série de conjuntos de células contíguas na direção Y. Cada conjunto de células na direção X é descrito por um número seqüencial, pelo número da linha onde se encontram as células, pela coluna da célula imediatamente à esquerda da primeira célula do conjunto, pela coluna da última célula do conjunto e por um valor que indica se a condição de fronteira na célula mais à direita do conjunto é por nível ou por velocidade. Da mesma forma, cada conjunto de células na direção Y é descrito por um número seqüencial, pelo número da coluna onde se encontram as células, pela linha da célula imediatamente abaixo da primeira célula do conjunto, pela linha da última célula do conjunto

e por um valor que indica se a condição de fronteira na célula mais acima do conjunto é por nível ou por velocidade. O arquivo com a descrição inicia, então, com o número de conjuntos de células na direção X, seguido da descrição de cada conjunto. Após, vem o número de conjuntos de células na direção Y seguido da descrição de cada conjunto. Assim, o arquivo de descrição de um domínio quadrado, indo da linha 5 à linha 7 e da coluna 5 à coluna 7, teria o seguinte formato:

```
3                          número de conjuntos de células na direção X
1  5  4  7  0              linha 5, vai da coluna 5 à coluna 7
2  6  4  7  0
3  7  4  7  0
3                          número de conjuntos de células na direção Y
1  5  4  7  0              coluna 5, vai da linha 5 à linha 7
2  6  4  7  0
3  7  4  7  0
```

O número seqüencial no início de cada linha e a condição de contorno no final da linha são mantidos apenas para compatibilidade com a primeiras versões do modelo, não sendo mais utilizadas, uma vez que as condições de contorno são fixas para o modelo, não dependendo do domínio sendo simulado.

- Estruturas de descrição do domínio: As informações lidas do arquivo de descrição do domínio são armazenadas em listas encadeadas mantendo a mesma estrutura do arquivo de descrição. Estas estruturas são utilizadas no modelo em todas as situações em que é necessário percorrer todas as células do domínio, como no particionamento, inicialização da malha, montagem das matrizes para o cálculo do nível em cada célula e interpolação da malha refinada.

- Malhas da hidrodinâmica e transporte 2D: Uma vez que cada processo tenha a descrição do subdomínio que irá manter, é alocada uma estrutura matricial para manter as variáveis correspondentes a cada célula do subdomínio. A estrutura utilizada para os modelos 2D, ilustrada na figura 4.2, é basicamente um vetor de apontadores para as linhas de células que compõem o domínio, onde cada linha do mesmo é alocada como um vetor. Para não precisar ser alocada a linha inteira a partir da coluna 0, é armazenado um vetor contendo a coluna inicial de cada linha, de forma que o vetor alocado para cada linha é apenas do tamanho necessário para armazenar as células entre a célula mais à esquerda e a célula mais à direita em cada linha. O custo adicional dessa decisão é que para cada acesso a um elemento é necessário subtrair-se do número da coluna buscada, o número da coluna inicial da linha, de modo a obter a posição relativa dentro do vetor. Assim, para acessar-se um elemento específico do domínio busca-se o apontador para a linha onde se encontra o elemento e, a partir do apontador obtido, subtraindo-se da coluna buscada o número da coluna inicial da linha, acessa-se o elemento buscado.

  Cada posição do vetor contém uma estrutura com as variáveis correspondentes a uma célula do domínio. Na malha da hidrodinâmica as principais variáveis são as componentes da velocidade da água nas direções X e Y, a profundidade e o nível da água acima do nível de referência naquele ponto. Além dessas variáveis são mantidas em cada célula diversas outras variáveis utilizadas na

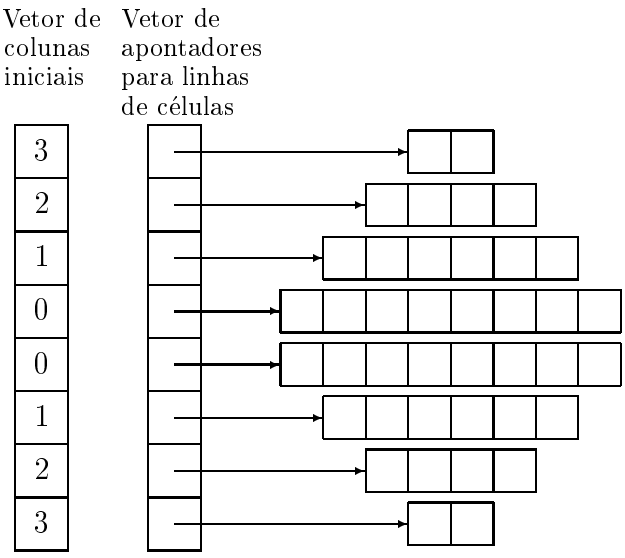


Figura 4.2: Estrutura de armazenamento das malhas 2D

montagem das matrizes de coeficientes. Para a malha de transporte são armazenados, para cada célula, a concentração de coliformes na célula, um termo de fonte, as componentes das velocidades nas direções X e Y, interpoladas a partir das componentes da malha da hidrodinâmica, além de outras variáveis também utilizadas para a montagem das matrizes de coeficientes.

- Malhas da hidrodinâmica e transporte 3D: Para o modelo 3D, cada posição do vetor das linhas corresponde a uma coluna de água (sendo chamada, por isso, de célula-base da coluna) e contém, além de algumas variáveis relativas a toda a coluna, um apontador para um vetor onde cada posição corresponde a uma célula do domínio 3D e contém as variáveis correspondentes à célula. A figura 4.3 mostra a estrutura de armazenamento do domínio 3D.

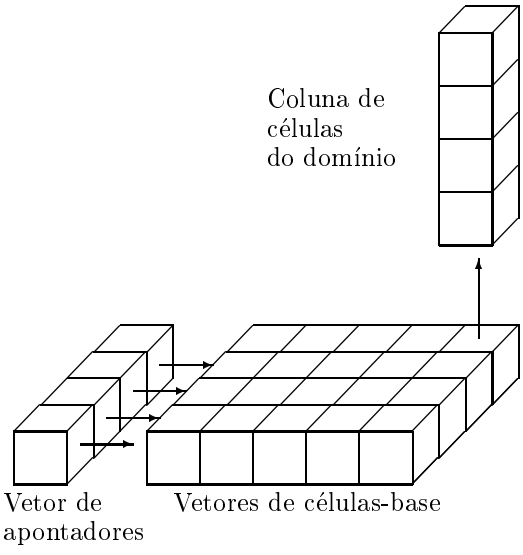


Figura 4.3: Estrutura de armazenamento das malhas 3D

4.1.4 Descrição algorítmica do modelo desenvolvido

Nessa seção será apresentada uma visão em um nível alto de abstração dos passos seguidos na execução dos diferentes modelos desenvolvidos. O procedimento aqui descrito é comum aos diversos modelos e as diferenças entre eles são apresentadas à medida que for necessário para a compreensão dos mecanismos implementados. O procedimento de inicialização, comum aos diversos modelos, é mostrado na figura 4.4.

```
Procedimento de inicialização

  Se processo=0
      Le arquivo de descrição do domínio
      Executa algoritmo de particionamento
      Monta estêncil e região de sobreposição para cada subdomínio
      Envia descrição do subdomínio para cada processo
  senao
      recebe descrição do seu subdomínio
  fim se
  Aloca malha da hidrodinâmica de seu subdomínio
  Inicializa variáveis e condições de contorno da malha
  Se processo=0
      Aloca malha do transporte baseado na posição dos emissores
      Inicializa variáveis e condições de contorno da malha
  Fim se
```

Figura 4.4: Procedimento de inicialização

Após efetuado o procedimento de inicialização, começa a execução dos ciclos de simulação propriamente ditos. O laço de execução dos ciclos é basicamente o mostrado na figura 4.5.

```
Laço Principal:

  t_hidro=0
  t_transp=0
  Enquanto numero_de_ciclos_h<CICLO_MAX
      Se t_hidro≤t_transp
          Executa ciclo da hidrodinâmica
          t_hidro = t_hidro + Delta_t_hidro
          numero_de_ciclos_h = numero_de_ciclos_h + 1
      senão
          Executa ciclo do transporte
          t_transp = t_transp + Delta_t_transp
          numero_de_ciclos_t = numero_de_ciclos_t + 1
      Fim se
  Fim enquanto
```

Figura 4.5: Laço principal do HIDRA

Em cada ciclo da hidrodinâmica é gerado, dependendo da abordagem utilizada pelo modelo, um ou vários sistemas de equações cuja solução resulta nos valores do nível de água para cada célula do domínio, e a partir do qual são calculadas as componentes da velocidade da água, também para cada célula do domínio.

Quando é utilizado o paradigma de particionamento de dados, é gerado um único sistema de equações, onde cada processo gera a parte do sistema que corresponde

ao seu subdomínio e o sistema é resolvido por um *solver* paralelo.

Quando é utilizado o paradigma de decomposição de domínio, cada processo gera um sistema correspondente ao seu subdomínio acrescido de uma região de sobreposição aos subdomínios vizinhos, e os sistemas são resolvidos. É feita uma troca de dados entre os processos detentores de subdomínios vizinhos, correspondentes às regiões de sobreposição, e é gerado um novo sistema para cada subdomínio e resolvido. Isso se repete até que as soluções locais dos sistemas convirjam para a solução global. Esse foi o paradigma utilizado até o momento nas versões do HIDRA com suporte para subdomínios irregulares. Uma explicação detalhada da abordagem de decomposição de domínio pode ser encontrada em (RIZZI, 2002).

O uso de decomposição de domínio como *solver* deu lugar, nos últimos anos, a uma abordagem híbrida, chamada Krylov-Schwarz, onde um *solver* paralelo, usualmente da classe dos métodos do subespaço de Krylov, como o Gradiente Conjugado ou GMRES, é acelerado por um pré-condicionador gerado pelo método de decomposição de domínio de Schwarz. A incorporação dessa abordagem ao HIDRA, utilizando diversos pré-condicionadores, é o tema de uma dissertação de mestrado sendo desenvolvida no grupo.

Assim, o ciclo da hidrodinâmica para a abordagem da decomposição de domínio pode ser descrito pelo algoritmo mostrado na figura 4.6.

```
Ciclo da hidrodinâmica:

  Repita // Decomposição de domínio
      Monta sistema de equações correspondente a subdomínio+sobreposição
      Resolve sistema de equações
      Atualiza nível em todo o subdomínio
      Troca valores de nível da região de sobreposição com vizinhos
  Até que critério de convergência seja atingido
  Calcula velocidades para todo o subdomínio
  Troca velocidades da região de sobreposição com vizinhos
  Calcula velocidade vertical para todo subdomínio (para modelo 3D)
  Se (resto(ciclo_h,15)=0)
      redistribui carga sem transporte
  Fim se
Fim Ciclo Hidrodinâmica
```

Figura 4.6: Ciclo da Hidrodinâmica

A redistribuição da carga sem transporte ocorre quando, após a redistribuição da malha do transporte e das células subjacentes da hidrodinâmica, a malha da hidrodinâmica fica desbalanceada. O procedimento de redistribuição de carga sem transporte é descrito no capítulo 6.

De forma alternada com os ciclos da hidrodinâmica, ocorrem os ciclos do transporte, cujo algoritmo básico é mostrado na figura 4.7.

A redistribuição da carga com transporte ocorre quando a alteração da malha do transporte fizer com que a mesma fique distribuída de forma desbalanceada entre os processos. Essas alterações podem ser devidas ao crescimento de concentrações próximas à fronteira e conseqüente alocação de células, ou devidas à desalocação de parte da malha quando a concentração em algumas células cair abaixo de um determinado valor. O procedimento de redistribuição de carga com transporte é descrito no capítulo 6.

```
Ciclo do transporte:

  Interpola malha refinada
  Monta sistema de equações correspondente ao subdomínio
  Resolve sistema de equações
  Atualiza concentrações em todo subdomínio
  Se (resto(ciclo_h,15)=5)
     Se há alguma concentração alta próxima à fronteira
        Expande a malha do transporte
        Busca células da hidrodinâmica necessárias para a malha expandida
     Fim se
  Fim se
  Troca dados da região da fronteira com os vizinhos
  Se (resto(ciclo_h,15)=10)
     redistribui carga com transporte
  Fim se
Fim Ciclo Transporte
```

Figura 4.7: Ciclo do Transporte

4.1.5 Métodos numéricos de resolução de sistemas de equações utilizados

Para a resolução dos sistemas de equações gerados pelo modelo desenvolvido neste trabalho e em (RIZZI, 2002) foram pesquisados e implementadas versões seqüenciais e paralelas de diversos métodos de solução. A necessidade de implementar diversos métodos vem do fato que alguns métodos são mais apropriados para alguns tipos de sistemas de equações. Existem diversas bibliotecas com rotinas para resolução de sistemas de equações, mas no início do desenvolvimento do modelo não se tinha conhecimento de bibliotecas que explorassem o paralelismo intranodos através do uso de *threads*, razão pela qual se decidiu pela programação dos métodos dentro dos modelos ao invés da utilização de bibliotecas. A partir da decisão de implementar os resolvedores, diversos trabalhos foram desenvolvidos pelo grupo sobre paralelização de resolvedores utilizando troca de mensagens e *threads*. Alguns dos trabalhos são (DORNELES et al., 2000), (PICININ, 2001) e (MARTINOTTO, 2001).

Nessa seção será apresentada apenas uma descrição sucinta dos resolvedores implementados nos modelos. Justificativas para a escolha de cada método e descrições dos métodos podem ser encontradas em (RIZZI, 2002).

A primeira versão do modelo 2D de hidrodinâmica utilizava um esquema de discretização ADI (*alternating directions implicit*) de direções alternadas, no qual no primeiro meio passo de tempo o esquema era implícito na direção X e explícito na direção Y e no segundo meio passo de tempo o esquema era explícito na direção X e implícito na direção Y. Utilizando esse esquema de discretização eram geradas matrizes tridiagonais (uma matriz para cada linha ou coluna do domínio) para cuja solução foi implementado um resolvedor que utilizava o método de Thomas, que é uma particularização do método de eliminação de Gauss para matrizes tridiagonais. Esse método é composto de duas fases. Na primeira fase a diagonal inferior é eliminada e a diagonal principal é normalizada, tal como na triangularização da matriz na eliminação de Gauss. Na segunda fase é feita uma retrosubstituição sendo gerada a solução do sistema.

O método de Thomas, assim como a eliminação de Gauss não é um método facil-

mente paralelizável devido à dependência entre as operações. É possível, entretanto, obter um bom grau de paralelismo quando diversos sistemas devem ser resolvidos utilizando um procedimento devido a Povitsky (POVITSKY, 1998). Nesse procedimento, cada sistema é gerado de forma distribuída entre vários processadores. Cada processador executa a primeira fase do algoritmo sobre a sua parte do sistema e quando termina passa os dados necessários para que o próximo processador execute a primeira fase sobre sua parte, enquanto o primeiro processador inicia a solução do próximo sistema. Dessa forma, os diversos processadores podem trabalhar simultaneamente sobre os diversos sistemas acarretando ganho de desempenho. Este método foi utilizado nas primeiras versões do modelo e uma descrição do mesmo pode ser encontrada em (DORNELES et al., 2000) e (RIZZI, 2002).

A versão seguinte do modelo utilizava um esquema de discretização e partições retangulares que resultavam em matrizes 5-diagonais SDP, para os quais o método de Thomas não era apropriado. Para a resolução dessas matrizes foi implementada uma versão seqüencial do método do gradiente conjugado baseada em (SHEWCHUCK, 1994), utilizada nas versões baseadas em decomposição de domínio. Além da versão seqüencial, também foi implementada uma versão paralela da mesma, onde a paralelização era baseada na paralelização das operações algébricas do método. Essa versão paralela é utilizada nas versões do modelo baseada em particionamento de dados.

Para melhorar a convergência do algoritmo do GC, foram implementados alguns precondicionadores para o mesmo. Entretanto, como as matrizes geradas pelo esquema de discretização da hidrodinâmica são razoavelmente bem condicionadas, a redução no número de iterações obtida não compensava o custo de calcular o precondicionador. Atualmente estão sendo desenvolvidos no grupo estudos sobre métodos de pré-condicionamento para o GC e para o GMRES, descrito a seguir.

Para a incorporação da capacidade de trabalhar com subdomínios irregulares no HIDRA, era necessário que os *solvers* utilizados pudessem operar sobre as matrizes irregulares resultantes desses subdomínios. Para isso os *solvers* passaram a utilizar como estrutura de dados para armazenamento da matriz um formato em que, para cada linha da matriz, eram armazenados a coluna e o valor dos elementos não nulos da mesma. Atualmente esse formato foi substituído pelo CSR (*Compressed Sparse Row*), um formato de armazenamento bastante popular para o armazenamento de matrizes esparsas (SAAD, 1996).

A incorporação do modelo de transporte de poluentes ao HIDRA fez surgir a necessidade de *solvers* que pudessem operar sobre as matrizes não simétricas resultantes do esquema de discretização utilizado. O método escolhido, e do qual foi implementada uma versão seqüencial, foi o GMRES, algoritmo do resíduo mínimo generalizado, proposto por Saad em 1986. Uma descrição do algoritmo do GMRES pode ser encontrada em (CATABRIGA, 1998) e um estudo discutindo a paralelização do mesmo com troca de mensagens e *threads* pode ser encontrado em (MARTINOTTO, 2001).

4.1.6 Desempenho dos *solvers* utilizados

As operações de maior custo computacional no método do Gradiente Conjugado são os produtos matriz-vetor, que quando aplicados a matrizes densas, apresentam complexidade quadrática na ordem da matriz. As matrizes geradas no HIDRA, entretanto, são matrizes esparsas com no máximo 5 elementos por linha, o que torna

a complexidade do produto matriz-vetor linear na ordem da matriz. Esse resultado torna também a complexidade do GC por iteração, linear no tamanho da matriz. Da mesma forma o GMRES, quando aplicado a matrizes esparsas estruturadas como as geradas no HIDRA, apresenta um comportamento linear na ordem da matriz. A complexidade da geração das matrizes de coeficientes dos sistemas também é linear no tamanho do domínio, uma vez que o número de operações para a montagem da matriz é aproximadamente igual para cada célula.

Para confirmar essas estimativas de complexidade, foram efetuadas algumas medidas com diversos tamanhos de domínio. A tabela 4.1 mostra, para 200 ciclos da hidrodinâmica, o tempo total (em segundos) gasto na montagem da matriz da hidrodinâmica, o tempo total utilizado pelo Gradiente Conjugado, e o tempo total usado pelo GMRES, utilizado como *solver* para a hidrodinâmica para essa medida de tempo.

Tabela 4.1: Tempo de montagem da matriz e execução dos *solvers*

Dimensões do Domínio	Tempo de Montagem da Matriz	Número de iterações do GC	Tempo do Solver GC	Número de iterações do GMRES	Tempo do Solver GMRES
10 x 10	1.33	373	0.075	371	1.126
20 x 20	6.60	403	0.26	379	5.17
30 x 30	13.62	421	0.61	384	16.841
40 x 40	23.38	434	1.13	381	32.63
50 x 50	34.94	570	2.20	385	49.04

Pelos dados apresentados pode-se constatar, conforme esperado, que o tempo de montagem da matriz é linear no número de células do domínio. Em relação ao tempo utilizado pelos *solvers*, tanto o Gradiente Conjugado quanto o GMRES também apresentam um tempo total por iteração linear no tamanho do domínio, havendo, para o Gradiente Conjugado, um aumento no número de iterações para atingir a convergência.

4.1.7 Descrição do ambiente utilizado

Os modelos computacionais foram implementados em linguagem C, utilizando o compilador gcc 2.91.60 sobre o sistema operacional Linux 2.2.1. Como biblioteca de troca de mensagens foi utilizado o MPICH versão 1.2.1, uma implementação do padrão MPI 1.0 desenvolvida no Argonne National Laboratory (GROPP et al., 1996). Diversos *clusters* foram utilizados nas diferentes fases desse trabalho. A maior parte do desenvolvimento dos modelos deu-se em um *cluster* do Instituto de Informática da UFRGS que é constituído por onze PCs (onze nodos). A tabela 4.2 (CANAL, 2001), mostra a configuração das máquinas desse *cluster*.

Os nodos são interconectados por duas redes de comunicação: a Fast-Ethernet e a Myrinet (BODEN; AL., 1995), sendo que a comunicação dos programas paralelos pode acontecer em qualquer uma das redes. É utilizado um único sistema de arquivos para toda a rede.

Quando as primeiras versões funcionais dos modelos ficaram prontas, foram efetuados testes para avaliar a escalabilidade dos algoritmos de particionamento e esquemas numéricos utilizados. Para se avaliar a escalabilidade do modelo com maior número de nodos utilizou-se, a convite do prof. Dr. Renato Silva, o *cluster* do

Tabela 4.2: Configuração dos nodos do *cluster* do Instituto de Informática

Máquina	CPU	RAM	Rede
Scliar	Pentium II 266 MHz	128 MB	*Fast-Ethernet*
Verissimo	Dual Pentium Pro 200 MHz	128 MB	*Fast-Ethernet* e *Myrinet*
Quintana	Dual Pentium Pro 200 MHz	128 MB	*Fast-Ethernet* e *Myrinet*
Dionelio	Dual Pentium Pro 200 MHz	128 MB	*Fast-Ethernet* e *Myrinet*
Euclides	Dual Pentium Pro 200 MHz	128 MB	*Fast-Ethernet* e *Myrinet*
Ostermann	Pentium Pro 200 MHz	64 MB	*Fast-Ethernet* e *Myrinet*
Luft	Pentium Pro 200 MHz	64 MB	*Fast-Ethernet* e *Myrinet*
Michelle	Dual Pentium III 500 MHz	256 MB	*Fast-Ethernet e SCI*
Demi	Dual Pentium III 500 MHz	256 MB	*Fast-Ethernet e SCI*
Sharon	Dual Pentium III 500 MHz	256 MB	*Fast-Ethernet e SCI*
Kim	Dual Pentium III 500 MHz	256 MB	*Fast-Ethernet e SCI*

Laboratório Nacional de Computação Científica no Rio de Janeiro, formado por 32 nodos monoprocessados.

Com a aquisição, por parte do Instituto de Informática da UFRGS em convênio com a Dell Computadores, do *cluster* Labtec, um *cluster* de 20 nodos biprocessados, tornou-se possível o desenvolvimento e avaliação dos algoritmos de balanceamento dinâmico de carga para um número maior de processadores. Todos os testes cujos resultados são mostrados no capítulo 6 foram executados no *cluster* Labtec.

O *cluster* Labtec possui 20 nodos biprocessados ligados entre si por dois *switches* com 24 portas Fast Ethernet (100 Mbps) cada um e uma porta Gigabit Ethernet (1000 Mbps), utilizada para fazer a comunicação dos nodos com o servidor. Cada nodo do *cluster* é composto por uma placa Dual Pentium III 1.1 Ghz, com 1 Gb de memória RAM, um disco rígido SCSI de 18 Gb e uma placa de rede Gigabit Ethernet. O servidor de acesso ao *cluster* é um Dual Pentium IV Xeon 1.8 Ghz com 1 Gb de memória RAM, um disco rígido SCSI de 36 Gb e uma placa de rede Gigabit Ethernet. A figura 4.8 (PICININ, 2003) mostra a topologia do *cluster* LabTec.

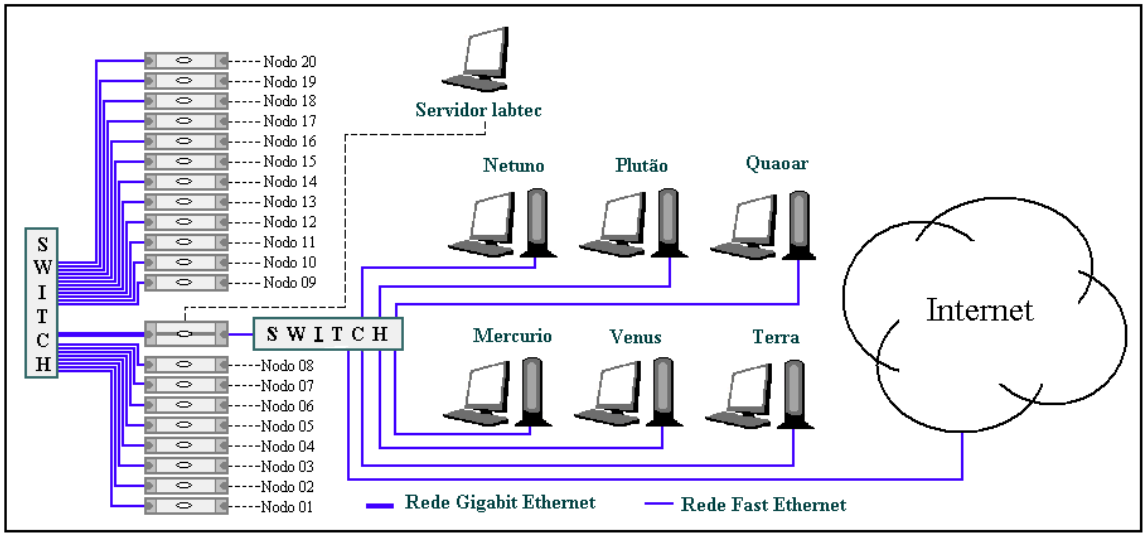


Figura 4.8: *Cluster* Labtec

4.1.8 Utilização de *threads* e troca de mensagens na mesma aplicação

O modelo de troca de mensagens é um entre os diversos modelos computacionais de paralelismo existentes, sendo o modelo mais utilizado em máquinas de memória distribuída. Ele pode ser utilizado também em máquinas multiprocessadas de memória compartilhada, havendo, entretanto, outros modelos que permitem explorar melhor os diversos processadores.

Em sistemas baseados no Unix, a abordagem tradicional é o uso de diversos processos rodando nos diferentes processadores. Um processo inclui um espaço de endereçamento e diversos recursos, além de uma (pelo menos) unidade de execução. Um processo é um mecanismo relativamente pesado e o chaveamento entre processos pode ter um custo elevado. O padrão POSIX alterou a definição de processo separando a unidade de execução do processo dos recursos do mesmo, passando um processo a ser definido como um espaço de endereçamento com uma ou mais linhas de execução, chamadas *threads*. Todas as *threads* dentro de um processo compartilham o mesmo espaço de endereço e recursos, o que faz com que o custo de chaveamento entre *threads* seja bem menor que o chaveamento de processos.

Além do padrão POSIX, existem outros padrões de *threads* como a definição de *thread* da Microsoft, utilizada na biblioteca Win32 e as *threads* da linguagem Java (CARISSIMI, 1999). Existem diversas ferramentas que permitem a aplicações a exploração de paralelismo por troca de mensagens e memória compartilhada. Entre elas pode-se citar o Athapascan0 (CARISSIMI, 1999) e o DECK (BARRETO et al., 2000). Canal (CANAL, 2001) apresenta uma implementação do método do Gradiente Conjugado utilizando o DECK onde foi utilizado paralelismo com troca de mensagens e *threads* e Picinin (PICININ, 2003) apresenta um estudo comparativo entre implementações dos algoritmos do Gradiente Conjugado e do Resíduo Mínimo Generalizado (GMRES) utilizando DECK, MPI e *threads*.

O padrão MPI-1.0 não especifica o modelo de execução de um processo MPI. Um processo MPI pode ser puramente seqüencial ou possuir diversas *threads*. Sobre isso o padrão simplesmente recomenda que as implementações do MPI permitam que diversas *threads* efetuem chamadas às primitivas do MPI de forma segura (CARISSIMI, 1999). Entretanto, a maior parte das implementações do MPI não permite a utilização segura de *threads*, devido à utilização, por parte delas, de outras bibliotecas que não suportem *threads*.

O padrão MPI-2.0 estabeleceu algumas definições em relação ao uso de *threads* em aplicações MPI. Para suporte a *threads* foi introduzida uma nova primitiva de inicialização, MPI_Init_thread, que define o nível desejado de suporte a *threads*. No nível *single* é permitida uma única *thread*. No nível *funelled*, são permitidas diversas *threads* mas apenas uma delas pode efetuar chamadas a primitivas do MPI. O terceiro nível é o nível *serialised*, no qual todas as *threads* podem efetuar chamadas a primitivas do MPI, mas não simultaneamente, sendo responsabilidade da aplicação a exclusão mútua. No quarto nível, o nível *multiple*, todas as *threads* podem chamar primitivas do MPI sem restrições. Cabe salientar que o padrão não permite endereçar *threads* individualmente.

Ainda não existe uma implementação completa do MPI-2.0, havendo apenas implementações de parte do padrão, principalmente em versões proprietárias. As versões de domínio público mais populares que são a MPICH e a LAM não permitem que mais de uma *thread* realizem chamadas simultâneas a primitivas do MPI.

É possível, entretanto, usar múltiplas *threads* juntamente com o MPICH se ape-

nas uma das *threads* utilizar chamadas do MPI (GROPP; LUSK, 2000). As outras não podem utilizar nenhuma primitiva do MPI, nem mesmo primitivas não envolvidas em comunicação como MPI_Wtime, devido à não reentrância do código do MPICH. Se apenas uma das *threads* utilizar chamadas do MPI, não ocorre problema de reentrância. Essa restrição vêm ao encontro do trabalho de Chen e Taylor (CHEN; TAYLOR, 1999), que sugerem que em arquiteturas do tipo *cluster* a concentração de toda a comunicação entre nodos em um único processador pode tornar mais eficiente a implementação.

Particularmente no Linux, o MPICH utiliza o `p4` que utiliza o `SIGUSR1`, mesmo sinal que é utilizado pela biblioteca `pthreads`. A partir da versão 1.2.0, na instalação do MPICH, é possível configurar o sinal que será utilizado (p.ex. `SIGUSR2`) pelo `p4`, para não utilizar o mesmo sinal que a biblioteca *pthreads*.

Nas diferentes versões dos modelos implementados nesse trabalho toda a comunicação entre processos é feita através de troca de mensagens. Trabalhos desenvolvidos no grupo (PICININ, 2001) e (MARTINOTTO, 2001) envolvendo a paralelização do método do gradiente conjugado e do GMRES com MPI e `pthreads` não apresentaram ganho de desempenho ao efetuar a comunicação entre processos residentes em processadores do mesmo nodo utilizando memória compartilhada. Uma possível razão para isso é o excessivo sincronismo inserido nos *solvers* na paralelização por *threads*. Trabalhos posteriores, entretanto, apresentaram um pequeno ganho de desempenho com a utilização de *threads*, sendo esse ganho tanto maior, quanto maior era o número de processos envolvidos (PICININ, 2003). Os resultados desses testes podem ser encontrados nos trabalhos citados. O GMCPAD continua desenvolvendo trabalhos na paralelização de solvers utilizando MPI e *threads*. *Solvers* utilizando *threads*, não foram incorporados ao modelo, mas os testes citados feitos no grupo mostram que é possível obter uma melhora do desempenho com o uso das mesmas.

Uma alternativa para o uso de *threads*, não explorada nesse trabalho, é a superposição de comunicação em uma das fases (hidrodinâmica ou transporte) e cálculo da outra fase. Uma vez feita a interpolação dos dados da hidrodinâmica para a malha do transporte, poder-se-ia proceder ao cálculo do transporte e do próximo ciclo da hidrodinâmica concorrentemente, de forma que quando uma das fases fosse efetuar trocas de mensagens, os processadores poderiam adiantar o cálculo da outra fase, diminuindo o tempo ocioso.

5 O MODELO DE PARTICIONAMENTO UTILIZADO NO HIDRA

Uma parte significativa deste trabalho consistiu na pesquisa, implementação e avaliação de algoritmos de particionamento de domínio. As primeiras versões dos modelos desenvolvidos utilizavam particionamento por faixas gerando subdomínios retangulares. À medida em que o nível de complexidade do modelo ia aumentando, com a utilização de estruturas de dados e *solvers* que permitiam a utilização de subdomínios não retangulares e matrizes estruturadas, porém irregulares, algoritmos de particionamento mais eficientes iam sendo incorporados ao modelo.

Neste capítulo será apresentado um rápido histórico dos diferentes particionamentos utilizados bem como alguns resultados obtidos. Uma descrição detalhada dos modelos e dos *solvers* utilizados pode ser encontrada em (RIZZI, 2002).

Como foi apresentado no capítulo 4, no modelo desenvolvido nesse trabalho e em (RIZZI, 2002) o cálculo da hidrodinâmica e do transporte de substâncias são efetuados separadamente. As equações que modelam ambos os fenômenos são tratadas de forma desacoplada, sendo que o cálculo do transporte de substâncias é feito utilizando os dados de velocidade e nível já calculados para a hidrodinâmica.

5.1 Particionamento da hidrodinâmica

Durante o desenvolvimento desse trabalho, diversas heurísticas de particionamento foram utilizadas. À medida que os *solvers* implementados para resolver os sistemas gerados eram aperfeiçoados, permitindo o uso de estruturas de dados mais flexíveis, novas heurísticas de particionamento eram utilizadas. O primeiro particionamento utilizado foi um particionamento por faixas, gerando subdomínios retangulares, onde células externas ao subdomínio eram acrescentadas ao mesmo para que os subdomínios resultantes tivessem uma geometria retangular e, conseqüentemente, fosse gerada uma matriz estruturada 5-diagonal para todo o subdomínio. Este acréscimo de células externas ao subdomínio acarreta cálculos desnecessários tendo como único ponto positivo a regularidade da matriz gerada, o que simplifica a implementação dos *solvers*. Essa abordagem foi abandonada tão logo foram desenvolvidos *solvers* para matrizes irregulares. A figura 5.1 (PICININ, 2003) mostra a divisão do domínio do Guaíba em 8 subdomínios retangulares utilizando esse método.

O particionamento por faixas tem a vantagem da simplicidade de implementação do particionamento, bem como resulta em fronteiras retas, o que faz com que cada linha do domínio resida inteiramente em um processador, o que é vantajoso ao usar o método ADI, utilizado nas primeiras versões do modelo. Além disso, o número

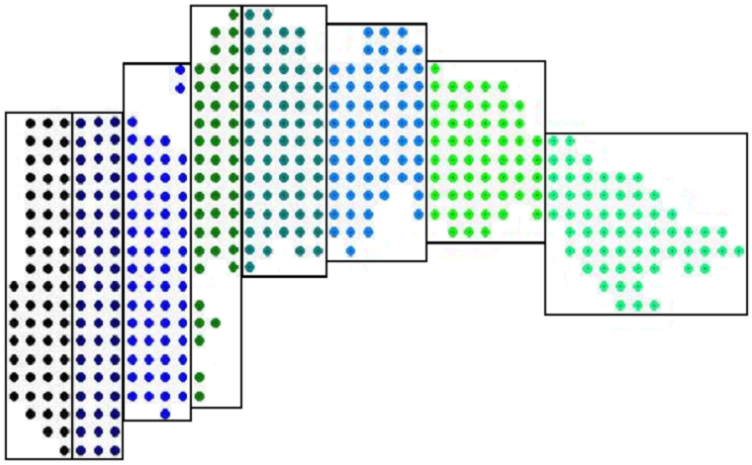


Figura 5.1: Particionamento do domínio do HIDRA em 8 subdomínios retangulares

reduzido de subdomínios vizinhos faz com que o número de mensagens trocadas seja pequeno, o que pode ser vantajoso no caso de mensagens pequenas, onde o tempo de latência é significativo em relação ao tempo total da mensagem. Entretanto, o particionamento por faixas retas pode resultar em um particionamento bastante pobre, podendo haver uma diferença grande no número de células de um subdomínio para o outro, além de gerar subdomínios vizinhos com uma fronteira larga, o que resulta em aumento na comunicação necessária. O particionamento em faixas com fronteiras não retas não resolve o problema das fronteiras grandes, mas melhora a distribuição das células, garantindo que a diferença no número de células entre dois sobdomínios vizinhos não seja maior que uma célula. O resultado da aplicação desse particionamento no domínio do Guaíba pode ser visto na figura 5.2 (PICININ, 2003).

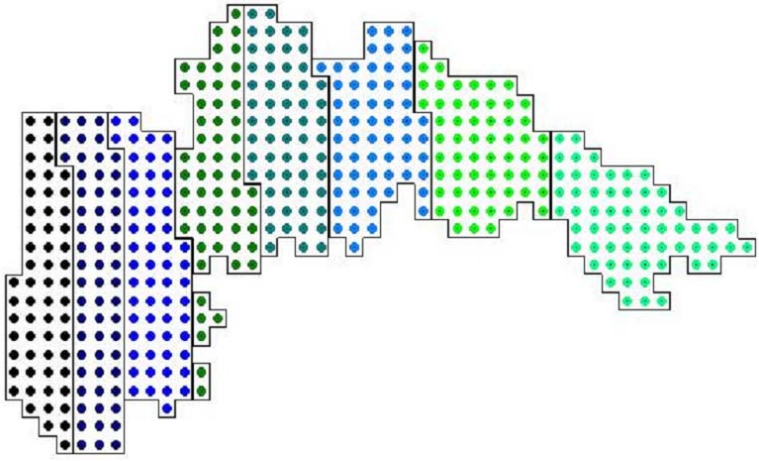


Figura 5.2: Particionamento em faixas com fronteiras não retas do domínio do HIDRA

O particionamento por faixas também obrigava que as fronteiras dos subdomínios fossem paralelas às direções X e Y, excluindo a possibilidade de cortes que não sejam paralelos a esses eixos, o que eliminava métodos como o inercial ou espectral. Uma família de algoritmos que se adaptava bem à restrição de subdomínios retangulares

é a bissecção coordenada recursiva (RCB) e algumas de suas variantes. Três versões desse algoritmo foram implementadas e aplicadas ao HIDRA para particionar o grafo do domínio computacional. Nas três versões, chamadas de SRCB (*Straight* RCB), a partir de um domínio inicial retangular são efetuadas bissecções com fronteiras retas, o que não garante um perfeito balanceamento mas gera subdomínios retangulares. A diferença entre as três versões implementadas do RCB reside no critério utilizado para a seleção da direção do corte e da forma de contagem do número de células em cada lado do corte.

A figura 5.3 mostra a divisão do domínio para 16 processadores empregando-se bissecção onde a direção do corte é perpendicular ao maior eixo do subdomínio sendo particionado e gerando partições com fronteiras retas. Uma descrição das diferentes implementações do RCB pode ser encontrada em (DORNELES et al., 2000)

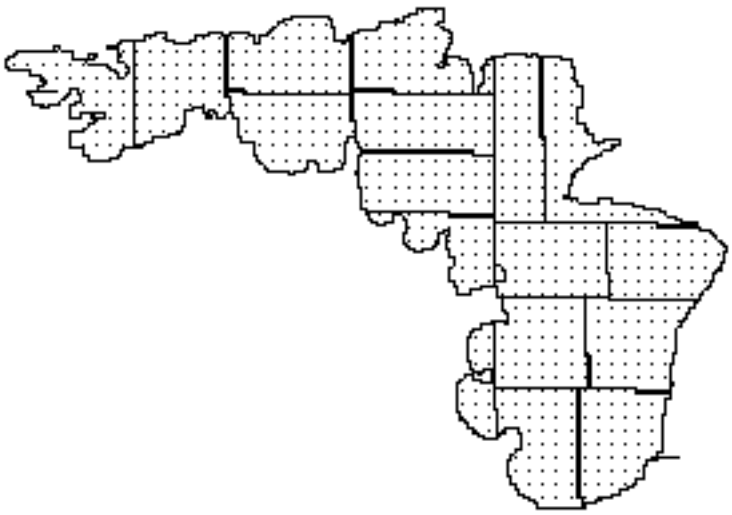


Figura 5.3: Particionamento do domínio do HIDRA para 16 processadores empregando-se o algoritmo SRCB

O algoritmo RCB na forma geral escolhe uma direção e um ponto onde será feita a bissecção do domínio e o divide. As três heurísticas utilizadas para a seleção da direção do corte e do ponto (qual a linha ou coluna) onde seria feita a bissecção são descritas a seguir:

Na primeira heurística era verificado, a cada bissecção, qual o maior eixo do subdomínio e o corte era feito ortogonal ao eixo. Contava-se o número de células internas do domínio e o corte era feito de forma a minimizar a diferença no número de células entre os dois subdomínios gerados. Quando utilizados subdomínios retangulares, como nas versões iniciais do modelo, a ordem dos sistemas gerados para cada subdomínio e conseqüentemente o custo de processamento de cada subdomínio era proporcional à área total dos subdomínios retangulares, e não somente ao número de células internas do subdomínio. A heurística descrita considerava apenas as células internas, o que é correto ao ser usada uma discretização onde o modelo seja resolvido apenas para as células internas, como no método ADI, mas faz com que haja uma diferença grande na área dos subdomínios retangulares entre os processadores, causando desbalanceamento.

Com a utilização de métodos de discretização que gerassem um único sistema para cada subdomínio retangular, foram implementadas duas outras heurísticas. A

primeira delas utilizava também o maior eixo do subdomínio sendo o corte proporcional a esse eixo. Definida a direção do corte, o ponto de corte era selecionado de forma a minimizar a area do maior entre os dois subdomínios resultantes. A terceira heurística (SRCB3), ao invés de utilizar o maior eixo, verifica nas direções X e Y o melhor ponto de corte e efetua o corte na direção de melhor distribuição de carga. Em nosso modelo, essa heurística resultou em um particionamento com um balanceamento ligeiramente melhor, apesar de gerar fronteiras maiores entre os subdomínios, o que implica maior comunicação.

As três heurísticas descritas para o particionamento geravam subdomínios retangulares com fronteiras retas, condição necessária para a geração dos sistemas de equações resultantes da discretização, uma vez que os *solvers* implementados, devido à estrutura utilizada para armazenar os sistemas, eram limitados a matrizes tridiagonais e pentadiagonais.

O desenvolvimento de *solvers* para matrizes irregulares e de rotinas para o gerenciamento de fronteiras com qualquer geometria permitiu o uso de subdomínios irregulares e, conseqüentemente, aumentou enormemente as possibilidades de uso de algoritmos de particionamento, permitindo, por exemplo, as trocas de vértices que ocorrem nos algoritmos de refinamento como o Kernighan-Lin ou o Fiduccia-Mattheyses, que não poderiam ser utilizadas com os subdomínios retangulares.

Tendo incorporado ao modelo a possibilidade de utilizar subdomínios não retangulares, o algoritmo RCB foi alterado para permitir a geração de fronteiras não retas, o que garante uma diferença de no máximo uma célula entre subdomínios vizinhos.

O algoritmo RCB apresentou resultados satisfatórios em relação à qualidade da partição gerada (corte de arestas e relação entre área dos subdomínios e perímetro) e foi o particionador utilizado para o particionamento da hidrodinâmica nas versões subseqüentes do modelo. Além dele, e para ter uma alternativa de particionamento, o particionador da biblioteca MÉTIS foi incorporado ao modelo e foram efetuados testes entre os dois. Os resultados obtidos com esses testes encontram-se no capítulo 7.

5.1.1 Estruturas de dados para gerenciamento das fronteiras irregulares

Um dos aspectos mais críticos no desenvolvimento dos modelos aqui descritos foi o correto gerenciamento das fronteiras entre os subdomínios. Durante a simulação, para manter a continuidade e consistência nos dados próximos às fronteiras, cada processo utiliza, para geração das matrizes correspondentes aos seus subdomínios, dados das células próximas às fronteiras dos subdomínios vizinhos, e para manter esses dados atualizados, a cada passo de tempo, cada processo deve efetuar uma troca de dados com os processos que mantêm os subdomínios vizinhos. Para poder efetuar essa troca, cada processo P_i gera e mantem estruturas que descrevem quais os seus subdomínios vizinhos e quais as células que deve enviar para os vizinhos e receber dos mesmos. Cada processo, para poder gerar a matriz correspondente a seu subdomínio, necessita de células de todos os subdomínios com que faz fronteira ou que possuem células próximas a ele. Essas células dos subdomínios podem estar em duas regiões distintas do subdomínio vizinho: o estêncil e a região de sobreposição. A região de sobreposição é composta pelas células que estão a uma distância menor ou igual a um valor pré-definido da fronteira entre os dois subdomínios e a região de estêncil é formada pelas células que estão a até uma célula de distância da região

de sobreposição.

A figura 5.4 mostra a região de fronteira entre dois processos P_i e P_j e as regiões de estêncil e sobreposição. Na figura, as células numeradas de 1 a 36 pertencem ao subdomínio do processo P_i e as células de 37 a 72 pertencem ao subdomínio do processo P_j. Se for considerada uma região de sobreposição de largura igual a 2, as células de 25 a 36 pertencem à região de sobreposição de P_i em P_j e as células de 13 a 24 pertencem ao estêncil de P_i em P_j. A escolha da largura da região da sobreposição deve ser feita com cuidado. Se a região for muito larga, um número excessivo de células será calculado por mais de um processo reduzindo o grau de paralelismo e a eficiência. Se a região for muito estreita, a convergência do método de decomposição de domínio será mais demorada. Nos testes realizados com o modelo, a largura de 2 células apresentou bons resultados. Um aumento na largura não causou melhora imediata na convergência do método e uma redução da largura piorou sensivelmente a convergência.

				P_i		P_j					
11	**12**	**23**	**24**	**35**	**36**	**47**	**48**	**59**	**60**	**71**	**72**
9	**10**	**21**	**22**	**33**	**34**	**45**	**46**	**57**	**58**	**69**	**70**
7	**8**	**19**	**20**	**31**	**32**	**43**	**44**	**55**	**56**	**67**	**68**
5	**6**	**29**	**30**	**17**	**18**	**41**	**42**	**53**	**54**	**65**	**66**
3	**4**	**15**	**16**	**27**	**28**	**39**	**40**	**51**	**52**	**63**	**64**
1	**2**	**13**	**14**	**25**	**26**	**37**	**38**	**49**	**50**	**61**	**62**

Figura 5.4: Região do estêncil e sobreposição

A região de sobreposição é utilizada nos modelos em que é utilizado, para paralelização, o método de decomposição de domínio de Schwarz com sobreposição (RIZZI et al., 2002) no qual cada subdomínio é estendido com um halo de células pertencentes aos subdomínios vizinhos e é gerado um sistema de equações para cada subdomínio estendido. As células pertencentes ao estêncil, por sua vez, são necessárias para fornecer os valores de contorno no cálculo dos coeficientes da matriz gerada para cada subdomínio, mas não há coeficientes associados a elas na matriz. Uma descrição detalhada do método de decomposição de domínio aditivo de Schwarz como utilizado nos modelos pode ser encontrada em (RIZZI, 2002). Um estudo comparativo sobre o uso diferentes métodos de decomposição de domínio nos modelos aqui apresentados está sendo desenvolvido como dissertação de mestrado por André Martinotto.

As estruturas que descrevem a fronteira (estêncil e sobreposição) são geradas durante o particionamento e consistem, para cada processo P_i, de uma lista de processadores P_j que mantém subdomínios vizinhos e, para cada processador P_j, as 4 estruturas a seguir:

- lista das células internas do processo P_j que estão a uma distância menor ou igual a 2 da fronteira entre os subdomínios ou, quando for utilizada sobreposição entre os subdomínios, lista das células que estão a uma distância menor

ou igual a 2 da área de sobreposição. Esta região é chamada de estêncil externo do processo P_j no processo P_i. Na figura 5.4 é formada pelas células de 49 a 60;

- lista das células internas do processo P_i que estão a uma distância menor ou igual a 2 da fronteira entre P_i e P_j ou, se houver sobreposição entre os subdomínios, lista das células internas que estão a uma distância menor ou igual a 2 células da região de sobreposição dos subdomínios dos processos P_j e P_i. Esta região é chamada de estêncil interno do processo P_i no processo P_j. Na figura 5.4 é formada pelas células de 13 a 24;

- lista das células internas de P_i que pertencem à região de sobreposição entre P_i e P_j. Esta região é chamada de sobreposição interna de P_i em P_j;

- lista das células internas de P_j que pertencem à região de sobreposição entre P_i e P_j. Esta região é chamada de sobreposição externa do proceso P_j no processo P_i.

5.1.2 Descrição do particionamento inicial

Em todos os modelos desenvolvidos, o particionamento do domínio é centralizado no processo 0 que busca em um arquivo a descrição do domínio e cria uma estrutura em memória que descreve sua configuração de linhas e colunas, conforme estrutura descrita na seção 4.1.3, bem como as condições de contorno. Sobre essa estrutura é executado o particionamento, seja utilizando o algoritmo RCB, a biblioteca MÉTIS, particionando por faixas ou lendo de um arquivo externo as informações sobre as partições e são geradas, para cada subdomínio, estruturas semelhantes à estrutura que descrevia o domínio original. Cabe salientar que esse particionamento é feito utilizando apenas as estruturas que descrevem a configuração do domínio, não tendo ainda sido alocadas as células dos domínios, o que poderia inviabilizar o particionamento de um domínio muito extenso por um único processo.

Após efetuado o particionamento, o processo 0 calcula, para cada subdomínio, as regiões de sobreposição e estêncil externos e internos e as envia para o processo responsável por cada subdomínio. De posse dessas estruturas que descrevem seu subdomínio, cada processo, incluindo o processo 0, aloca memória para todo o seu subdomínio e inicializa as variáveis de cada célula do domínio após o que é iniciada a simulação.

5.2 Particionamento da malha refinada do transporte de substâncias

Conforme foi colocado no capítulo 2, a simulação do transporte de substâncias exige uma malha de resolução maior que a hidrodinâmica. Uma alternativa para compatibilizar a construção de modelos computacionalmente eficientes com a necessidade de alta resolução local é o uso de malhas refinadas localmente. Refinamento local é uma estratégia que permite que a malha seja concentrada em subregiões onde ocorrem maiores atividades. Nessas subregiões a malha pode ser ajustada e refinada para obter soluções acuradas, sem o custo computacional de uma malha globalmente refinada. Além disso, como a pluma de contaminantes pode tomar várias configu-

rações e direções à medida que o tempo evolui, a malha refinada deve ser construída de forma que ela acompanhe o evento à medida que esse se desenvolva.

A malha refinada utilizada nesse trabalho para o transporte de substâncias é equiespaçada e é construída através de processos de interpolação espacial e temporal considerando que a solução das ESWs fornece os valores de u, v e η , que são, em cada ponto da malha e em cada nível de tempo, dados de entrada da ETM. A descrição do processo de interpolação pode ser encontrada em (RIZZI, 2002). Assim, para malhas regulares o refinamento local pode ser obtido construindo subdomínios encaixados no domínio maior. As figuras 5.5 a 5.8 mostram o comportamento de uma pluma durante 10000 ciclos. Nessa seqüência de figuras a fonte de poluentes emite durante 500 ciclos. A figura 5.5 mostra a situação após 50 ciclos, no início da formação da pluma, quando a mesma contém apenas as células muito próximas ao ponto de emissão. À medida em que os emissores despejam poluente, a concentração de poluente na região próxima aos emissores aumenta, e, através de advecção e difusão, aumenta a região onde a concentração de poluentes é significativa.

Da mesma forma como novas células são acrescentadas à malha quando a concentração nos limites ultrapassa um determinado ponto, sempre que a concentração na fronteira cair abaixo de um determinado valor as células onde a concentração de poluente é desprezível são desalocadas reduzindo a área da malha refinada. Isso ocorre, por exemplo, quando um emissor de poluente deixa de emitir e a advecção desloca a região contaminada para longe do emissor, restando, na região próxima ao mesmo uma concentração muito baixa. A desalocação das células desta região pode reduzir de forma significativa a carga computacional, uma vez que o custo computacional de processar cada célula do domínio (excluídas as células da fronteira) é o mesmo, independentemente da concentração que possua. A figura 5.6 mostra a situação após 500 ciclos, no momento em que o emissor de poluente cessa de emitir. A partir desse momento, apesar de não haver mais emissão a pluma continua se expandindo por efeito da difusão e da advecção, como é mostrado na figura 5.7, que mostra a pluma após 7600 ciclos. Entretando, apesar da área da pluma estar maior as concentrações são menores por efeito da dispersão e do decaimento de forma que a partir de um certo ponto a concentração em algumas regiões da malha cai abaixo do valor mínimo para desalocação (nesta simulação foi utilizado um valor igual a 10) e partes da malha são desalocadas, como é mostrado na figura 5.8.

Nas primeiras versões do HIDRA o particionamento da malha onde ocorre o transporte acompanhava o particionamento estático da hidrodinâmica, ou seja, cada célula da malha refinada residia no processador onde residia a célula da hidrodinâmica associada àquela célula da malha do transporte. Nessas versões, o processo de montagem e manutenção do particionamento da malha refinada de transporte de substâncias é como se segue: no início da simulação o processo 0 envia para cada processo as informações sobre os emissores de poluentes presentes em cada subdomínio da hidrodinâmica, e aqueles processos que possuem emissores em seu subdomínio alocam uma malha refinada na região próxima ao emissor. À medida em que a pluma de poluente vai crescendo, a malha refinada vai sendo aumentada. Ao chegar à fronteira entre dois subdomínios, o processo responsável pelo subdomínio vizinho recebe a informação e aloca a malha refinada para que o transporte do poluente ocorra também no subdomínio vizinho.

Esse esquema de particionamento funciona bem durante a fase de hidrodinâmica, garantindo um bom balanceamento de carga durante a mesma. Entretanto, durante

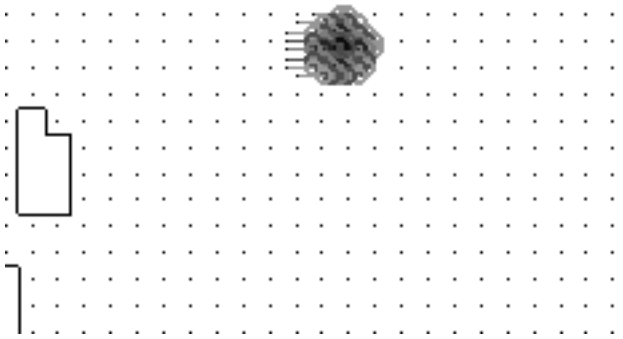


Figura 5.5: Situação da malha refinada após 50 ciclos

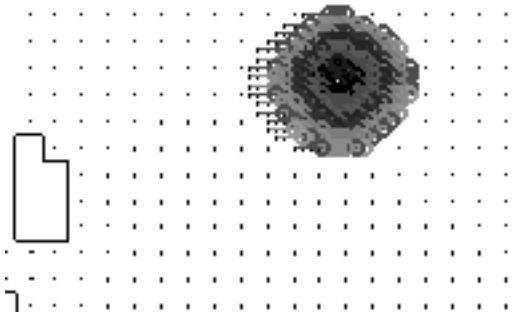


Figura 5.6: Após 500 ciclos

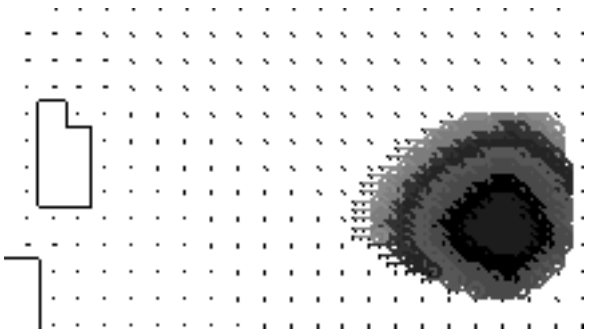


Figura 5.7: Após 7600 ciclos

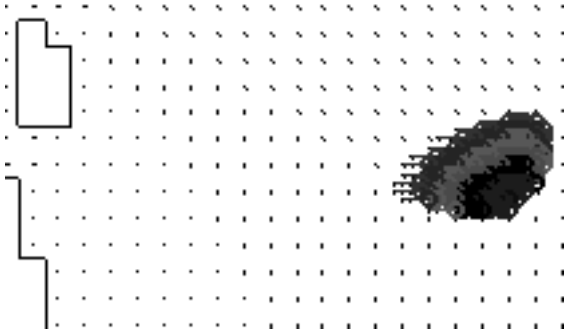


Figura 5.8: Após 9300 ciclos

os ciclos de transporte, o desbalanceamento pode ser enorme, podendo ficar concentrada toda a malha refinada em um único processador, de forma que todos os outros processadores ficarão ociosos durante a fase de transporte. Esse desbalanceamento foi confirmado em medidas feitas com o modelo quando se constatou que os ciclos de transporte ocupam mais tempo de CPU que os ciclos de hidrodinâmica, mesmo que o transporte ocorra apenas em uma região pequena da região onde ocorre hidrodinâmica. Isso ocorre devido ao refinamento maior da malha de transporte e ao maior custo computacional do cálculo do transporte para cada célula, quando comparado ao custo computacional do cálculo de uma célula da hidrodinâmica. Esses resultados mostraram a necessidade do desenvolvimento de mecanismos de balanceamento dinâmico para a malha do transporte. Esses mecanismos também foram objeto deste trabalho e estão descritos no próximo capítulo.

6 O MODELO DE BALANCEAMENTO DINÂMICO DE CARGA NO HIDRA

Uma das características do modelo desenvolvido é a utilização de balanceamento dinâmico de carga durante a simulação. A necessidade de balanceamento dinâmico de carga decorre principalmente do modelo de transporte de poluentes do HIDRA, no qual o fenômeno ocorre apenas sobre uma parte do domínio, parte essa que varia ao longo dos ciclos de simulação, o que pode causar um desbalanceamento de carga. O domínio onde ocorre a hidrodinâmica é estático e os mecanismos de particionamento estático descritos no capítulo anterior são suficientes para a maior parte das situações. A incorporação de características de alagamento e secamento ao modelo de hidrodinâmica, quando células podem ser acrescidas ou retiradas da malha, dependendo do nível da água em cada ponto do domínio durante a simulação, acarretaria em alterações à malha e poderia criar a necessidade de mecanismos de balanceamento dinâmico para a hidrodinâmica. Tais características, entretanto, não fazem parte do modelo implementado, no qual pressupõe-se que o domínio permanece inalterado durante toda a simulação.

Mecanismos de balanceamento dinâmico de carga no modelo da hidrodinâmica poderiam ser utilizados também em um ambiente onde os nodos de processamento fossem compartilhados com outras aplicações. Nesse ambiente, dependendo da carga atual de cada nodo, partes do domínio poderiam migrar dos processos mais carregados para os menos carregados de forma a manter a carga computacional dos nodos equilibrada. Os mecanismos estudados nessa seção podem ser utilizados para melhorar o desempenho do modelo de hidrodinâmica em tal contexto.

Uma solução para o problema de balanceamento dinâmico de carga em problemas multi-fase como o abordado nesse trabalho pode ser obtida de diversas maneiras:

1. Através do particionamento da malha da hidrodinâmica sem levar em conta o transporte. O particionamento da malha refinada é feito baseado na malha menos refinada de forma que cada célula da malha refinada é mantida no mesmo processo que a célula da hidrodinâmica à qual ela está associada. Essa abordagem, que não pode ser considerada uma abordagem de balanceamento dinâmico, não possui o custo adicional do rebalanceamento, mas pode levar a um forte desbalanceamento na fase de transporte de substâncias;

2. Através do particionamento de ambas as malhas independentemente e da transferência, no início de cada ciclo do transporte de substâncias, dos dados da hidrodinâmica necessários a cada célula da malha refinada, a partir do processo que os detêm. Essa abordagem assegura um balanceamento ótimo

em ambas as fases, mas a migração dos dados da hidrodinâmica pode ser alta;

3. Através do particionamento da malha refinada de transporte e das células da malha da hidrodinâmica associadas a ela, ou seja, as células onde ocorre transporte, mantendo no mesmo processo as células do transporte e as células da hidrodinâmica associadas a elas. Após isso, particiona-se o restante da malha da hidrodinâmica, sobre a qual não ocorre transporte. Esta é a abordagem mais complexa, envolvendo o maior número de trocas de mensagens e que tem o maior custo, mas assegura um bom balanceamento em ambas as fases e reduz a comunicação às fronteiras, apesar de possivelmente aumentar o tamanho das mesmas.

As três abordagens foram implementadas no HIDRA e nesse capítulo são descritas em detalhes as três soluções implementadas bem como os resultados obtidos. Para cada abordagem será descrito o processo de expansão da malha do transporte, seguido dos mecanismos de balanceamento implementados. Na descrição dos mesmos será utilizado o termo "abordagem das malhas não acopladas"para a segunda abordagem, na qual as células de transporte não ficam necessariamente no mesmo processo que as células da hidrodinâmica correspondentes e o termo "Balanceamento em duas fases", para a terceira abordagem, na qual cada célula da malha de transporte fica no mesmo processo que a célula da hidrodinâmica associada.

A métrica utilizada para avaliar a carga nas diversas implementações, tanto na fase de hidrodinâmica quanto na fase de transporte de substâncias, é o número de células do subdomínio. Essa escolha se justifica pela complexidade linear da montagem das matrizes e da execução dos *solvers* que fazem com que o tempo para a execução de um ciclo de simulação para um subdomínio seja diretamente proporcional ao número de células, conforme foi mostrado na seção 4.1.6. Outras escolhas possíveis para a avaliação da carga são discutidas na seção 6.4.

6.1 Particionamento do transporte baseado na hidrodinâmica

Nessa abordagem, que não é propriamente um mecanismo de balanceamento dinâmico de carga e que é descrita neste capítulo apenas para comparação com os dois métodos de balanceamento propostos, é feito o particionamento da hidrodinâmica no início da simulação e cada célula da malha de transporte residirá no processo que mantém a célula da hidrodinâmica associada.

6.1.1 Expansão da Malha do Transporte

Nas três abordagens implementadas, o desbalanceamento de carga inicia na malha do transporte de substâncias, quando a concentração de alguma célula próxima à fronteira externa da malha do transporte cresce além de um determinado nível. Quando isso ocorre a malha é expandida alocando-se novas células. Estas novas células podem ser alocadas no mesmo processo onde ocorreu a concentração alta ou, se a concentração alta ocorrer em uma célula próxima à fronteira, em um processo que detenha um subdomínio vizinho. O algoritmo da figura 6.1 resume o procedimento do recálculo da malha:

Nessa abordagem não há migração de células da hidrodinâmica e as novas células de transporte são alocadas no processo que detem a célula da hidrodinâmica associada. A passagem da pluma de poluente entre subdomínios vizinhos é feita através

```
para cada célula interna e do estêncil da malha de transporte:
    se concentracao está acima de CONCENT_MAX
        se alguma celula vizinha é uma célula de contorno então
            aloca célula vizinha para a malha refinada
```

Figura 6.1: Algoritmo de recálculo da malha de transporte

do estêncil do transporte. Quando a concentração alta ocorre em uma célula de transporte na fronteira entre dois processos a célula de concentração alta é enviada para o processo vizinho como estêncil e em volta dela são alocadas novas células.

A continuidade da pluma de poluente na fronteira entre processos é garantida pelo esquema de discretização, que é explícito na horizontal e implícito na vertical, sendo gerados sistemas independentes de equações para cada coluna vertical do domínio.

Pode ocorrer que, ao longo do tempo, a concentração de poluente em algumas regiões da malha fique tão baixa que deixe de ter influência no comportamento da pluma. Isso pode ocorrer, por exemplo, quando um emissor de efluente cesse de emitir e a pluma seja deslocada, pelo efeito da advecção, da região em torno do emissor. Como o custo computacional de processar cada célula independe da concentração de poluente nas mesmas, o custo de processar estas células pode ser considerável e a remoção das mesmas da malha pode melhorar o desempenho do modelo. Esse mecanismo foi implementado na versão do modelo que utiliza a primeira abordagem e está previsto para implementação nas outras duas abordagens. As figuras 5.5 a 5.8, apresentadas e descritas na seção 5.2, ilustram o crescimento da malha e a desalocação de partes da mesma.

6.2 Abordagem das malhas não acopladas

Nesta abordagem, o particionamento da hidrodinâmica é o particionamento estático da inicialização, não havendo nenhum mecanismo de balanceamento dinâmico para a malha da hidrodinâmica, e o balanceamento dinâmico do transporte é feito de forma independente. Ao começar um ciclo de transporte, é feita a interpolação das variáveis da malha da hidrodinâmica para as variáveis da malha do transporte. Para cada célula do transporte são calculadas suas variáveis a partir da célula da hidrodinâmica à qual ela está acoplada. Cada processo detem uma parte da malha da hidrodinâmica, definida no particionamento inicial, e uma parte da malha do transporte, pela simulação das quais será responsável. Além disso, cada processo deverá manter alocadas as células da hidrodinâmica sobre as quais ocorra transporte, mesmo que elas pertençam a um subdomínio residente em outro processo. Os valores das variáveis associadas a essas células serão buscados quando for necessário. Se a célula da hidrodinâmica reside no mesmo processo que a célula do transporte, ela já está disponível ao processo e basta efetuar a interpolação. Se a célula da hidrodinâmica reside em outro processo, ela é buscada para efetuar a interpolação. A estrutura do ciclo de transporte nesse modelo é apresentada na figura 6.2. Basicamente a cada ciclo de transporte verifica-se se o ciclo anterior era um ciclo de hidrodinâmica ou de transporte. Se o ciclo anterior foi de hidrodinâmica, o campo de velocidades na malha de hidrodinâmica foi alterado e os dados da hidrodinâmica necessários para a atualização da malha refinada devem ser buscados nos processos que os detêm. O mesmo pode ocorrer se houve expansão da malha de transporte,

situação onde podem ser necessários, para calcular o campo de velocidades da malha refinada, dados da hidrodinâmica que ainda não estavam disponíveis. Após os dados da hidrodinâmica serem buscados, é feita a interpolação espacial e temporal da malha não refinada, gerando o campo de velocidades e nível da água da malha refinada. Após gerada a malha refinada, são gerados os sistemas de equações para o cálculo das concentrações de poluentes no próximo passo de tempo e os sistemas são resolvidos. Após calculadas as concentrações em todo o subdomínio, as concentrações nas fronteiras são trocadas entre os processos vizinhos. A cada 10 ciclos é feito o recálculo da malha, ou seja, verifica-se se há células próximas à fronteira com concentração alta e a malha deve ser expandida, ou se há células com concentração excessivamente baixa que devem ser desalocadas. Esse procedimento não é efetuado em todos os ciclos de transporte porque envolve troca de mensagens e o custo de efetuá-lo em todos os ciclos seria excessivo. A redistribuição de carga é feita a cada 30 ciclos, uma vez que pode acarretar um custo adicional significativo de troca de mensagens e as alterações na carga, causadas pelo aumento da malha, são lentas.

```
Ciclo transporte

se ciclo anterior era de hidrodinâmica ou
     houve recálculo da malha então
       troca células de hidrodinâmica necessárias para o transporte
fim se
interpola malha refinada
resolve malha do transporte
troca halos com processos vizinhos
se ((ciclo de transporte mod 10)==0) então
       recalcula malha
fim se
se ((ciclo de transporte mod 30)==0) então
       redistribui carga
fim se
fim Ciclo transporte
```

Figura 6.2: Ciclo de transporte no modelo das malhas não acopladas

6.2.1 Expansão da Malha do Transporte

A expansão da malha de transporte nessa abordagem é semelhante à da abordagem anterior, ou seja, quando a concentração de alguma célula próxima à fronteira cresce além de um determinado nível a malha é expandida alocando-se novas células. Nessa abordagem, entretanto, a decisão de em qual processador será alocada a nova célula é baseada no particionamento por faixas da malha de transporte, e não na malha de hidrodinâmica, como ocorre na abordagem anterior.

O algoritmo da figura 6.3 resume o procedimento do recálculo da malha:

No recálculo da malha de transporte verifica-se para cada célula próxima à fronteira da malha se a concentração de poluente está acima de um determinado valor. Se estiver, são alocadas células em volta da célula com concentração alta, de forma que a pluma de poluente possa expandir-se.

O particionamento usado no transporte de substâncias foi um particionamento por faixas de forma que as coordenadas da célula determinam em que processador ela será alocada. Se as células da hidrodinâmica associadas às células recém alocadas pertencem a outro processo, as células são colocadas em uma lista de células que

```
para cada célula interna e do estêncil da malha de transporte:
    se concentracao está acima de CONCENT_MAX
        se alguma célula vizinha é uma célula de contorno da malha refinada então
            aloca célula vizinha para a malha refinada
            se célula da hidrodinâmica associada pertence a outro processo então
                coloca célula na lista de células a buscar
            fim se
        fim se
    fim se
fim para
```

Figura 6.3: Algoritmo de recálculo da malha de transporte

serão buscadas a cada novo ciclo de transporte. Este crescimento da malha refinada é o que inicia o desbalanceamento. O estado inicial de um domínio com uma mallha refinada próxima a dois pontos de emissão (em cima à esquerda e a área retangular aproximadamente no centro) é mostrado na figura 6.4. Nesse domínio o campo inicial de velocidades é nulo em todo o domínio, há uma entrada de água na parte superior esquerda do domínio e uma saída de água no canto inferior direito. A figura 6.5 mostra uma imagem mais detalhada da situação inicial da malha refinada, onde o emissor de poluente está situado no canto inferior esquerdo da malha refinada e a situação da mesma após 300 ciclos da hidrodinâmica. Pode-se observar o crescimento da malha refinada de maneira uniforme ao redor do ponto de emissão. Isso ocorre porque, como nos ciclos iniciais o campo de velocidade é nulo em todo o domínio, o crescimento da malha refinada ocorre na maior parte devido à difusão.

Figura 6.4: Lago Guaiba com dois pontos de emissão

6.2.2 Busca de informações da Hidrodinâmica

Ao alocar uma nova célula de transporte, o processo responsável deve buscar as variáveis da hidrodinâmica necessárias para a interpolação das variáveis correspondentes na célula de transporte recém alocada. Para isso é necessário que ele saiba qual processo mantém os dados da hidrodinâmica correspondentes à célula recém alocada. Essa informação pode ser obtida a partir do número do processo que contém a célula da hidrodinâmica associada à célula do transporte que tem a

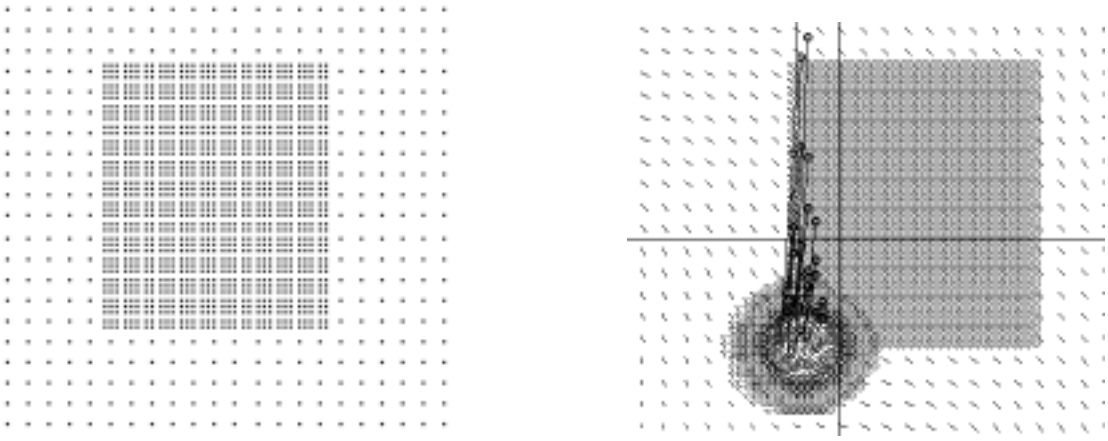


Figura 6.5: Situação inicial da malha refinada e após 300 ciclos

concentração alta e que foi causadora da alocação da nova célula. Ao alocar a célula do transporte, o processo envia uma mensagem ao processo que contém os dados da hidrodinâmica da célula causadora da alocação da nova célula. Se esse processo não for o responsável pela célula solicitada, ele envia uma mensagem para o processo que a detém, para que passe a enviar a célula para o processo que a solicitar.

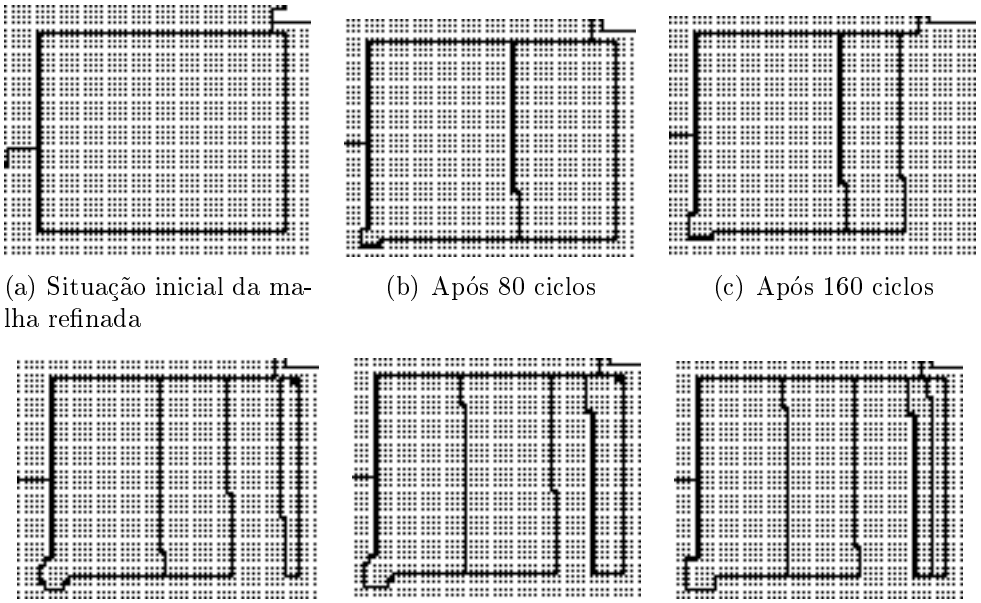
Cada processo mantém uma lista das células da hidro que possue e que devem ser enviadas para outros processos ao terminar um ciclo de hidrodinâmica e começar um ciclo de transporte. Se ocorrem dois ciclos de hidrodinâmica sem um ciclo de transporte entre eles, somente após o segundo ciclo, após o qual será executado um ciclo de transporte, os dados da hidrodinâmica devem ser enviados. Após enviados os dados da hidrodinâmica, é efetuada a interpolação dos mesmos para a malha refinada (nível e velocidades) e iniciados os ciclos de transporte. Durante a execução dos ciclos de transporte, novas células refinadas podem ser alocadas e podem ser necessárias informações sobre a hidrodinâmica de células não residentes no processo. Para evitar que isso ocorra é mantido um estêncil de células da hidrodinâmica de forma que o aumento da malha do transporte sempre ocorre sobre células da hidrodinâmica residentes no processo em questão. Quando, durante o crescimento da malha do transporte, uma célula do estêncil da malha da hidrodinâmica é alcançada, é necessário redefinir o estêncil da malha da hidrodinâmica, alocando novas células da malha da hidrodinâmica e recebendo, no início dos ciclos do transporte, os dados correspondentes também às células do novo estêncil.

6.2.3 Particionamento em Faixas

O particionamento adotado para o transporte nessa abordagem e na abordagem em duas fases é o particionamento por faixas (*Stripwise Partitioning*), descrito na seção 2.2.1. Neste esquema, a cada processo é atribuída uma faixa de colunas, de forma que cada processo N saiba, ao alocar novas células para sua pluma, se as células recém alocadas devem ser processadas por ele (o processo N) ou pelo processo vizinho (N-1) ou (N+1). Dessa forma é definida uma fronteira "virtual"entre os processos simplificando o gerenciamento das fronteiras. Para isso, cada processo mantém a informação da coluna inicial e da coluna final de seu subdomínio do transporte. A distribuição das faixas do domínio entre os processos será feita de forma que faixas vizinhas pertençam a processos de numeração contígua e, portanto, o processo N

troque dados com os processos N-1 e N+1.

Inicialmente, como a quantidade de células onde ocorre transporte é pequena, todo o domínio do transporte é colocado no processo 0. Na inicialização o processo 0 envia para cada processo P_i as coordenadas das células da malha da hidrodinâmica que P_i deve enviar para o processo P_0 para a interpolação. À medida em que a pluma cresce e o processamento do transporte começa a causar uma diferença significativa na carga entre processos vizinhos, as fronteiras entre os subdomínios do transporte são deslocadas e as células que trocam de subdomínio em conseqüência dessa alteração na fronteira são transferidas entre os processos responsáveis pelos subdominíos envolvidos. A diferença de carga a partir da qual deve ocorrer a redistribuição depende de fatores como o tempo necessário para efetuar a redistribuição e da velocidade com que ocorrem as alterações no domínio e uma discussão sobre ela se encontra na seção 6.2.5. A figura 6.6 ilustra a distribuição da malha refinada entre 6 processos nos 400 primeiros ciclos. A figura 6.6(a) mostra a situação inicial, quando a malha refinada está toda no processo 0. Após alguns ciclos o processo 0 envia uma parte da malha para o processo 1, alcançando a situação mostrada na figura 6.6(b). As figuras 6.6(c) a 6.6(f) mostram a distribuição da malha refinada respectivamente após 160, 240, 320 e 400 ciclos de transporte.



(a) Situação inicial da malha refinada (b) Após 80 ciclos (c) Após 160 ciclos

(d) Após 240 ciclos (e) Após 320 ciclos (f) Após 400 ciclos

Figura 6.6: Comportamento da malha ao longo de 400 ciclos

Como foi apresentado na seção 2.2.1, apesar de o particionamento por faixas apresentar um balanceamento de carga ótimo, em que o número de células de subdomínios vizinhos difere em no máximo uma célula, o corte de arestas gerado por ele pode estar longe do ótimo, sendo a qualidade do corte fortemente dependente da geometria do domínio. Por essa razão, na escolha do esquema de particionamento da malha do transporte avaliou-se também uso de um particionamento irregular, como ocorre com a malha da hidrodinâmica. Entretanto, em um particionamento irregular, cada processo teria que manter uma descrição completa da geometria dos subdomínios pertencentes a todos os outros processos, para verificar, quando a malha fosse expandida, se as células recém alocadas já não faziam parte de algum

subdomínio. Esse problema não ocorre com o particionamento por faixas, onde as coordenadas da célula recém alocada já definem o processo que será responsável por ela, razão que levou a adotar, para o modelo, o particionamento por faixas.

6.2.4 Freqüência de disparo do mecanismo de balanceamento

Como foi mostrado na seção 3.1.2, um ponto importante a ser considerado em esquemas de redistribuição de carga é a escolha do momento mais apropriado para a redistribuição e a freqüência com que a mesma deve ser feita, uma vez que a redistribuição pode ter um alto custo computacional associado. A freqüência ótima de disparo do mecanismo de balanceamento é função de diversos fatores como o tempo para efetuar a redistribuição da carga e a taxa de variação da carga nos processadores. Em ambientes onde as variações de carga são pequenas, a verificação da necessidade de redistribuição da carga pode ser feita a intervalos maiores. O tempo para efetuar o rebalanceamento pode ser visto como tendo um custo fixo, devido à latência das mensagens envolvidas no balanceamento, que independe da quantidade de carga a ser movida, e um custo variável, proporcional à quantidade de carga transferida.

O processo de redistribuição de carga, como implementado nesse trabalho, pode ser dividido em duas fases. Na primeira fase os processos trocam informações com seus vizinhos sobre a carga de cada um e, na segunda fase, é efetuada a troca de carga.

Os mecanismos implementados utilizam apenas comunicação entre processadores que detenham subdomínios vizinhos, tanto na fase de troca de informações sobre carga quanto na fase de transferência de células e cada uma das fases ocorre simultaneamente em todo o domínio. Para isso, é utilizada uma freqüência fixa para troca de informações sobre carga, como por exemplo, a cada 50 ciclos de transporte. Após a troca de informações sobre carga, se houver diferença significativa entre a carga de um processo e o processo vizinho, é disparado o processo de transferência de carga. Como a troca de células entre dois processos pode ter efeitos em outros processos que não tenham participado diretamente da troca, devido à dependência que os mesmos possam ter das células migradas (p.ex. uma célula migrada pode pertencer ao estêncil externo do processo), é necessária uma terceira fase de troca para acertar as estruturas de sobreposição e estêncil dos processos vizinhos. Todos os processos participam dessa fase, tenham ou não participado da transferência de carga.

Em situações onde a malha tende a uma situação de estabilidade, com a carga bem equilibrada durante longos períodos, pode-se reduzir o custo da redistribuição reduzindo a freqüência de verificação. Para isso é necessário que todos os processos saibam se ocorreu, em qualquer um dos outros processos, alguma transferência de carga a cada ciclo de redistribuição de forma que possam alterar a freqüência de redistribuição de forma sincronizada. Para isso pode ser utilizada uma comunicação global onde todos os processos informem aos outros se ocorreu transferência. A partir dessa informação, todos os processos podem ajustar a freqüência de verificação aumentando-a se ocorreram transferências significativas de carga no período ou reduzindo-a se não ocorreram transferências (o que significa que o domínio alcançou uma configuração estável).

6.2.5 Reparticionamento da malha do transporte

No início da simulação toda a malha de transporte reside no processo 0. Durante a simulação, à medida em que a malha cresce, partes dela vão sendo migradas para o processo 1, após para o processo 2 e assim por diante, até que a malha esteja distribuída entre vários processos, como foi mostrado na figura 6.6.

A decisão de migrar células ou não é tomada a partir da diferença de carga existente entre cada processo e os processos vizinhos. Como uma parte do custo da transferência de carga entre processos independe do número de células transferidas (sendo função apenas da latência da rede e do número de mensagens trocadas), pode não ser vantajoso disparar o mecanismo de troca de carga quando a diferença é pequena. No HIDRA o valor mínimo de diferença de carga a partir do qual é disparado o mecanismo de balanceamento, chamado de DIF_CARGA_MAX_TRANSP, foi implementado como um parâmetro. Na seção 7.7 são apresentados os resultados de execuções do modelo com diferentes valores para esse parâmetro, de forma a avaliar seu efeito no desempenho do modelo.

Uma vez que a diferença de carga entre dois subdomínios vizinhos atinge um valor máximo, parte da carga é transferida entre os subdomínios. Um parâmetro a ser considerado e que afeta de forma significativa o desempenho do modelo é o percentual da diferença da carga que será transferido. Se o percentual for muito baixo, será necessário um tempo muito grande para atingir uma situação de boa distribuição de carga. Se o percentual for alto, pode-se chegar a uma situação de oscilação onde partes da carga fiquem sendo transferidas entre subdomínios vizinhos sem convergir para uma situação estável ou demorando muito tempo para isso. Esse valor foi implementado no HIDRA como um parâmetro chamado COEF_DIF_CARGA_TRANSP e o resultado de execuções com diferentes valores para esse parâmetro pode ser encontrado na seção 7.7

Além da diferença máxima de carga aceitável antes de disparar o mecanismo de balanceamento e do percentual da diferença a ser transferida, também é considerado, para o disparo da transferência de carga, se um dos dois subdomínios envolvidos na transferência não ficará abaixo de um número mínimo de células, seja por ter transferido células para o outro, seja por ser um domínio recém-criado e que ficará com um número de células abaixo desse mínimo. Se tal ocorrer a transferência de carga não é executada. Esse valor, chamado no modelo de TAM_MIN_TRANSP, também foi implementado na forma de um parâmetro.

Uma abordagem alternativa à difusão é fazer com que todos os processos troquem informações entre si sobre a carga de cada um e, de posse da informação da carga de cada nodo, calculem uma solução possível para o problema de transferência de carga. Se o particionamento do domínio é irregular, podem existir diversos esquemas diferentes de migração de carga que levem a uma distribuição equilibrada. Se é utilizado um particionamento em faixas, a solução para o problema de calcular a quantidade de carga que deve ser transferida entre cada processo é única. O cálculo da carga a ser transferida entre os processos pode ser centralizado em um único processo, que receberá informações de todos os processos sobre a carga de cada um, calculará uma solução possível e enviará para cada processo a quantidade de carga que deve trocar com cada vizinho. Essas soluções envolvem diversas comunicações globais que podem afetar a escalabilidade da solução. Essa solução foi implementada para efeito de comparação com a abordagem por difusão e os resultados obtidos podem ser encontrados na seção 7.9

6.2.6 Escolha das células a serem transferidas e transferência das mesmas

Uma vez calculada a quantidade de carga a ser transferida de um processo para outro, devem ser selecionadas as células que serão transferidas entre os processos. Como nessa abordagem é utilizado um particionamento por faixas para a malha do transporte, as células transferidas entre os processos são as células das colunas mais próximas à fronteira, deslocando a mesma.

A transferência das células é efetuada em diversas etapas. O processo que vai enviar a carga deve enviar as células que estão sendo transferidas e removê-las da estrutura que descreve o seu subdomínio. O processo que recebe a carga deve incorporar cada célula recebida à estrutura que descreve o seu subdomínio. Além dos processos diretamente envolvidos na troca de carga, os processos que possuem as células da malha da hidrodinâmica associadas às células de transporte que foram trocadas de processo devem estar cientes da troca para que passem a enviar as células da malha da hidrodinâmica para o processo que recebeu as células migradas. Para que o processo que possui as células da malha da hidrodinâmica subjacentes às células que migraram saiba para onde enviá-las é necessário que o processo que enviou (ou recebeu) as células o notifique da troca.

6.3 Balanceamento em duas fases

Nessa abordagem, o mecanismo para balanceamento dinâmico é executado em duas fases diferentes, cada fase ativada em diferentes momentos e envolvendo algum grau de sincronização e troca de mensagens entre processos que detêm subdomínios vizinhos. Quando uma nova célula é alocada para a malha de transporte de substâncias, a célula da hidrodinâmica correspondente àquela posição do domínio deve estar residindo, após o final do processo de recálculo da malha do transporte, no mesmo processo. Se ela pertencer a outro processo, ela é migrada, através de algumas trocas de mensagens, para o processo que possui a célula recém alocada. Esse processo, descrito na seção 6.3.1, pode causar um desbalanceamento na malha da hidrodinâmica.

Esse desbalanceamento na malha da hidrodinâmica pode ser compensado pela migração de células da hidrodinâmica sobre as quais não ocorre transporte de substâncias. Esse balanceamento separado do transporte e da hidrodinâmica pode resultar em mais de um subdomínio para cada processador onde cada processador terá uma parte proporcional do domínio da hidrodinâmica onde ocorre transporte e uma parte do domínio da hidrodinâmica onde o mesmo não ocorre, e essas duas partes podem não estar conectadas. Entretanto, o processamento de um domínio composto de diversas partes não conectadas entre si é feito da mesma forma que um domínio totalmente conexo, uma vez que o sistema resultante da discretização é gerado levando em conta somente as células internas, independentemente de pertencerem a um único domínio conexo ou não. Essa separação da malha da hidrodinâmica em duas partes pode acarretar um aumento significativo na quantidade de comunicação entre os processadores, devido ao aumento das fronteiras e ao aumento proporcional das regiões de sobreposição e estêncil. Deve ser avaliado se o ganho com o particionamento separado das duas malhas (transporte e hidrodinâmica) compensa o custo adicional do aumento de comunicação.

O mecanismo para verificar se a malha deve ser expandida é ativado a cada 20

ciclos. O número ótimo de ciclos ou mesmo se existe um número ótimo de ciclos ainda está em aberto e outras medidas devem ser feitas. Após alguns ciclos, as alterações na malha podem causar um forte desbalanceamento entre os processos e o procedimento de balanceamento dinâmico propriamente dito deve ser executado.

6.3.1 Expansão da Malha do Transporte

Nessa abordagem a expansão da malha de transporte é análoga à forma como é feita na abordagem das malhas desacopladas. A diferença maior reside em que, quando é alocada uma célula de transporte sobre uma célula da hidrodinâmica que não pertence ao processo em questão, essa célula é transferida do processo que a detém para o processo onde foi alocada a nova célula de transporte e passa a pertencer ao mesmo. O algoritmo da figura 6.7 resume o procedimento do recálculo da malha para essa abordagem:

```
para cada célula interna e do estêncil da malha de transporte faça
    se concentração está acima de CONCENT_MAX então
        se alguma célula vizinha é uma célula de contorno da malha refinada então
            aloca célula vizinha para a malha refinada
            se célula da hidrodinâmica associada pertence a outro processo então
                coloca célula na lista de células a buscar
            fim se
        fim se
    fim se
fim para
```

Figura 6.7: Algoritmo de recálculo da malha de transporte

Assim, no recálculo da malha de transporte verifica-se para cada célula próxima à fronteira da malha se a concentração de poluente está acima de um determinado valor. Se estiver, são alocadas células em volta da célula com concentração alta, de forma que a pluma de poluente possa expandir-se.

Após a expansão da malha, as células da hidrodinâmica associadas às células recém alocadas devem ser buscadas dos processos que as possuem. Essa transferência de células, devido à possibilidade de envolver vários processos que possuam células do estêncil ou da sobreposição das células transferidas exige a troca de diversas mensagens entre todos os processos envolvidos de forma a atualizar as estruturas de descrição do estêncil e sobreposição em todos os processos envolvidos. A seqüência de mensagens, executada por cada um dos processos, é descrita na figura 6.8.

A troca de mensagens para garantir a consistência entre as fronteiras ocorre também, com pequenas alterações, nas outras duas situações onde células são transferidas entre processos e é responsável por uma parcela significativa do custo do rebalanceamento. As duas situações de transferência de células ocorrem por ocasião da redistribuição de carga, na transferência de células da hidro onde ocorre transporte e na transferência de células da hidro onde o mesmo não ocorre.

O particionamento usado na malha de transporte foi um particionamento por faixas de forma que as coordenadas da célula determinam em que processador ela será alocada. Se as células da hidrodinâmica associadas às células recém alocadas pertencem a outro processo, as células são solicitadas ao outro processo e, após recebidas, passam a fazer parte do processo que detem as células do transporte recém alocadas.

```
Troca células da hidrodinâmica // Troca celulas da hidro que estão na
                                  // lista de células a buscar

envia para os processos vizinhos a lista de células que precisa receber
recebe dos processos vizinhos a lista de células que precisa enviar
para cada processo para quem precisa enviar células:
      monta lista de células internas a serem enviadas
      a partir da lista de células internas monta lista de células
          do estêncil e da sobreposição
fim para
atualiza o número do processo de todas as células que vão ser transferidas
para cada processo para quem precisa enviar células
      envia células internas
      envia células de estêncil
      envia células de sobreposição
fim para
para cada processo de quem precisa buscar células
      busca células internas, de estêncial e sobreposição
fim para
troca informações com processos vizinhos sobre células que deve passar a enviar
     para outros processos como estêncil ou sobreposição
atualiza estêncil e sobreposição internos
notifica vizinhos sobre alterações no estêncil e sobreposição internos
     para atualizar estêncil e sobreposição externos
corrige estêncil e sobreposição internos
```

Figura 6.8: Algoritmo de troca de células de hidrodinâmica

6.3.2 Reparticionamento da malha do transporte

O reparticionamento da malha de transporte é feito de forma similar ao reparticionamento no modelo com as malhas desacopladas valendo as considerações sobre os parâmetros feitas na seção 6.2.5. Há, entretanto, algumas diferenças entre os processos de transferência de carga nos dois modelos.

A primeira diferença se refere à seleção das células do transporte que serão transferidas. No modelo das malhas acopladas, cada célula da hidrodinâmica na qual ocorre transporte deve ser tratada pelo mesmo processador que trata das células de transporte que estão sobre ela. Assim, todas as células de transporte que estão sobre a mesma célula da hidrodinâmica devem ser tratadas pelo mesmo processo. Para que isso se mantenha, quando uma célula da malha refinada é transferida de um processo para o outro, por ocasião da transferência de carga, todas as células da malha refinada que estejam associadas à mesma célula da hidrodinâmica devem ser migradas juntas. Isso resulta em fronteiras como a mostrada na figura 6.9, nas quais a largura da descontinuidade é de uma célula da hidrodinâmica e não de uma célula do transporte, como ocorre na abordagem anterior. Assim, a quantidade mínima de células de transporte a serem enviadas de um processador para o outro é o número de células do transporte existentes em uma célula da hidrodinâmica, ou seja, REF_TRANSP^2 onde REF_TRANSP é o refinamento da malha do transporte. Na figura 6.9 foi utilizado REF_TRANSP igual a 3, fazendo com que o número mínimo de células a serem transferidas seja igual a 9 e a largura da descontinuidade na fronteira seja igual a 3.

Na abordagem por difusão cada processo compara sua carga com a carga de cada vizinho e se a diferença de carga estiver acima de algum valor, parte da carga

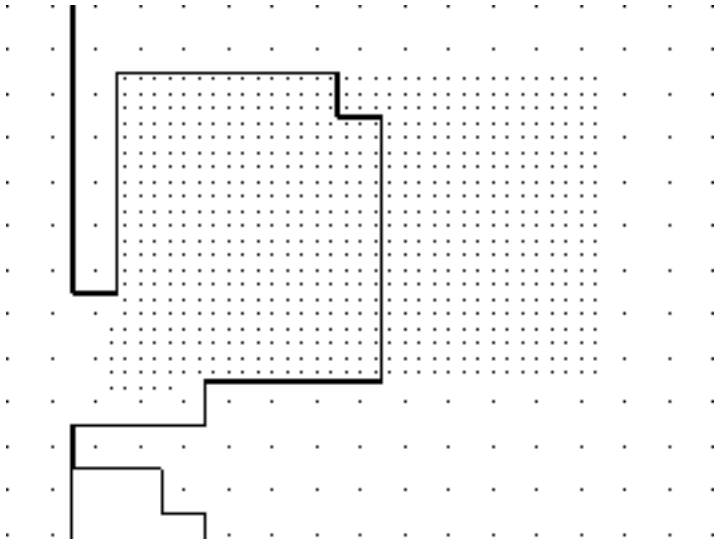


Figura 6.9: Fronteira no particionamento por faixas

é migrada para o vizinho. Numa outra abordagem é calculada a carga total de todo o domínio (dada, por exemplo, pelo número total de células), e a partir desse valor é calculada a carga esperada em cada processo (número de células em cada subdomínio), sendo efetuada então uma migração de células entre os subdomínios para chegar a uma distribuição homogênea. Se o particionamento do domínio é irregular, podem existir diversos esquemas diferentes de migração de carga que levem a uma distribuição homogênea. Se é utilizado um particionamento em faixas, a solução para o problema de calcular a quantidade de carga que deve ser transferida entre cada processo é única. A descrição do particionamento por faixas utilizado na abordagem das malhas desacopladas, na seção 6.2.3 também se aplica a essa abordagem.

Quando células da malha refinada são transferidas, as células correspondentes da malha da hidrodinâmica também devem ser transferidas a fim de que passem a residir no mesmo processo. Isso pode causar um desbalanceamento na malha da hidrodinâmica, que é resolvido na próxima fase do mecanismo de balanceamento dinâmico.

6.3.3 Particionamento da Hidrodinâmica

Para o balanceamento da parte da malha da hidrodinâmica onde não ocorre transporte, cada processo troca informações com todos seus vizinhos (considerando o particionamento irregular) e, de forma análoga ao que ocorre na malha do transporte, se a diferença está acima de um valor pré-determinado, a carga é transferida. Um problema adicional que não existe no balanceamento da malha do transporte é a seleção das células a serem transferidas.

A seleção das células a serem transferidas deve ser feita em cada uma das duas fases do balanceamento. A seleção das células do transporte a serem transferidas é análoga à abordagem anterior, uma vez que, para o transporte, também é utilizado o particionamento por faixas, onde as coordenadas de cada célula já definem a qual subdomínio a mesma pertence. Assim sendo, as células que serão transferidas entre dois processos são as células mais próximas à fronteira entre os mesmos. Quando são transferidas as células do transporte, as células da hidrodinâmica associadas também

são transferidas de forma que o processo responsável por uma região da malha de transporte também seja o processo responsável pelas células da hidrodinâmica associadas.

Na segunda fase, quando ocorre o balanceamento das partes da malha da hidrodinâmica onde não ocorre transporte, como é utilizado um particionamento irregular, deve-se buscar as células que minimizem o corte de arestas. Na primeira implementação do algoritmo não foi utilizada nenhuma heurística de seleção, baseando-se simplesmente na ordem seqüencial das mesmas dentro das estruturas que definiam os domínios, evitando as células que residissem em regiões utilizadas como sobreposição por mais de 2 processos, para reduzir o número de processos envolvidos e, conseqüentemente, a quantidade de comunicação. Esse critério gerou subdomínios com fronteiras bastante irregulares, com maior necessidade de processamento e comunicação (pela maior região de sobreposição resultante da maior fronteira).

A versão atual do modelo utiliza um algoritmo baseado no custo de cada célula, a partir do qual pode-se calcular o ganho obtido no corte de arestas ao enviar uma célula de um subdomínio para outro. Esse custo, de forma análoga ao calculado no algoritmo Kernighan-Lin, é dado pela diferença do custo externo da célula (número de células vizinhas pertencentes ao domínio destino) e do custo interno da célula (número de células vizinhas pertencentes ao mesmo domínio).

As células a ser enviadas são selecionadas entre as células da região de sobreposição uma vez que as células internas e do estêncil, por não fazer fronteira com o subdomínio vizinho, certamente aumentariam o corte de arestas. Uma vez que o primeiro critério a ser atendido por esse mecanismo é a distribuição equilibrada de carga, conjuntos de células podem ser transferidos mesmo que acarretem um aumento no corte de arestas.

O algoritmo utilizado para a seleção das células é mostrado na figura 6.10:

```
se (num_cel_sobre ≤ num_cel_a_enviar) então
      envia todas as células da sobreposição
senão
      cont=0
      calcula o custo de cada célula
      enquanto ( cont ≤ num cel a enviar )
          busca célula de maior custo
          marca célula como pertencente ao subdomínio destino
          coloca célula na lista de células a enviar
          atualiza custo das células vizinhas
          cont = cont + 1
      fim enquanto
fim se

onde num_cel_a_enviar é o número de células que
devem ser enviadas para o processo vizinho e num_cel_sobre é
o número de células da sobreposição que não tem transporte
```

Figura 6.10: Algoritmo de seleção de células

Esse algoritmo pode ser implementado com um custo linear no número de células da sobreposição. O cálculo do custo de cada célula se resume a testar em qual processo se encontra cada uma das células vizinhas e, utilizando-se uma estrutura do tipo *bucket* pode-se efetuar a busca da célula de maior custo e a atualização das células vizinhas com um custo constante. O algoritmo descrito foi incorporado ao

modelo mantendo as partições com um corte de arestas visivelmente de qualidade, ao longo de 15.000 ciclos de simulação.

É sabido que algoritmos de refinamento baseados no custo dos vértices como o Kernighan-Lin e o Fiduccia-Mattheyses permitem escapar de mínimos locais, ou seja, trocam células que pioram temporariamente o corte de arestas para, nas trocas seguintes, resultar em um corte melhor. Entretanto, esses algoritmos não garantem escolher a cada passo a célula que resulta no melhor corte, uma vez que esse problema é NP-difícil. Da mesma forma o algoritmo utilizado no modelo, apesar de ter melhorado a qualidade do corte, não garante o melhor corte. Uma forma de melhorar a qualidade do corte é calcular o ganho de cada célula considerando não apenas o custo da célula, mas também o ganho obtido com as trocas subseqüentes. Um algoritmo baseado nessa abordagem teria um custo alto mas poderia mostrar-se viável se as células a ser analisadas se restringirem às da região de sobreposição e se o número de trocas subseqüentes avaliadas para cada célula for pequeno.

6.4 Utilização dos Mecanismos em Ambientes Compartilhados e *Clusters* heterogêneos

Os mecanismos descritos nesse capítulo foram implementados e a maior parte dos testes foi efetuada em *clusters* homogêneos onde a única aplicação sendo executada era o HIDRA. Em ambientes onde outras aplicações compartilham os nodos com o modelo, a disponibilidade de processamento de cada nodo para a execução do modelo é reduzida quando o nodo passa a executar outras aplicações em concorrência com o modelo. Os mecanismos aqui descritos podem ser utilizados para deslocar carga dos nodos com menor disponibilidade de processamento para os nodos menos carregados de modo a tornar a distribuição de carga proporcional à disponibilidade de cada nodo. Da mesma forma, em *clusters* heterogêneos em relação à capacidade de processamento dos nodos, os mecanismos descritos podem ser utilizados para deslocar carga dos nodos de menor capacidade para os nodos de maior capacidade. Pode-se utilizar um particionamento estático considerando o *cluster* formado por nodos de mesma capacidade e, aplicando os mecanismos descritos, transferir carga de modo que cada nodo tenha carga proporcional à sua disponibilidade de processamento.

Para detectar essas situações o mecanismo para avaliação de carga utilizado até agora, que é a contagem do número de células em cada subdomínio, não é apropriado, uma vez que a partir do número de células do subdomínio não se pode estimar a disponibilidade de processamento do nodo para a aplicação. Uma forma de avaliar a relação entre a carga e a capacidade de processamento do nodo pode ser a medição do tempo de montagem da matriz de coeficientes e de resolução dos sistemas gerados para cada subdomínio ou simplesmente a medição do tempo de execução do *solver*. Um processo que leve um tempo maior que o processo vizinho para resolver o sistema resultante de seu subdomínio possui uma relação entre sua carga e sua disponibilidade de processamento maior e pode enviar células para o processo vizinho para melhor equilibrar a carga. Essa métrica para avaliação de carga aparece em (KWOK et al., 1999), onde é descrito um mecanismo de balanceamento de carga centralizado no qual, a intervalos de tempo pré-determinados, cada nodo envia a um nodo central, informações sobre o tempo necessário para processar a sua carga. O processador central irá calcular, então, a nova carga de cada processador de

acordo com o tempo necessário para processar a carga atual. Um balanceador de carga utilizando como medida de avaliação de carga o tempo de processamento do subdomínio foi implementado no HIDRA e resultados obtidos com ele podem ser encontrados na seção 7.8.

Os mecanismos aqui descritos também podem ser utilizados no balanceamento de carga em *clusters* heterogêneos, mesmo para aplicações onde os domínios são estáticos, como uma alternativa a um particionamento estático que leve em conta a capacidade de processamento dos nodos, não considerando diferenças na rede de interconexão. Para isso pode-se medir o tempo para a montagem e solução dos sistemas em cada processador e comparar-se com o tempo dos vizinhos ou com a média global para acionar o mecanismo de difusão ou balanceamento.

7 MEDIDAS E ANÁLISE DOS RESULTADOS

O desempenho das três abordagens descritas no capítulo anterior foi avaliado utilizando o *cluster* Labtec, descrito na seção 4.1.7.

Cada execução consistiu de uma simulação de 10.000 ciclos de transporte, período considerado suficiente para que o custo da distribuição inicial seja amortizado em toda a simulação. Todas as simulações foram feitas baseadas no domínio do Lago Guaíba originalmente com um espaçamento nas direções X (ΔX) e Y (ΔY) de 200 metros, chamado de Guaíba 200. Foram feitas medidas também utilizando domínios obtidos a partir do Guaíba 200 submetendo-o a um refinamento. Assim, o Guaíba 200 com um refinamento de 2 vezes gerou um domínio do Lago Guaíba com malha de 100 metros por 100 metros. Foram feitos testes utilizando para o particionamento o algoritmo RCB e o MÉTIS. Nos testes envolvendo a malha refinada de transporte, ela foi obtida aplicando-se um refinamento igual a 3 na malha da hidrodinâmica.

O domínio do Guaíba utilizado nos testes é o mostrado na figura 6.4. A malha é refinada inicialmente em duas regiões distintas, representadas na figura como as regiões mais escuras, havendo um emissor de poluentes no canto inferior esquerdo de cada uma das regiões refinadas.

7.1 Modelo sem balanceamento dinâmico

A tabela 7.1 mostra os resultados obtidos para o Guaíba 200 com refinamento igual a 2 (REF=2) resultando em um domínio com ΔX=ΔY=100 metros. Foi utilizado para o particionamento estático da hidrodinâmica o algoritmo RCB. Não foi utilizado balanceamento dinâmico para o transporte, ficando cada célula da malha do transporte alocada no processo responsável pela célula da hidrodinâmica sobre a qual a célula do transporte reside, o que já é definido no particionamento inicial. Nessa tabela são apresentados, para cada número de processos, o tempo total para os 10.000 ciclos, o tempo total utilizado na hidrodinâmica, incluindo o tempo necessário para as trocas de mensagens envolvidas, o tempo utilizado pelo *solver* da hidrodinâmica, o tempo total utilizado pelo transporte, novamente incluindo as mensagens trocadas, e o tempo utilizado pela rotina de montagem das matrizes de coeficientes para o transporte e pelo *solver*. Todos os tempos são dados em segundos e o valor entre parênteses ao lado de cada valor representa o valor relativo ao valor com um único processador (no caso dos tempos totais, representa o *speedup*). Essa estrutura da tabela será mantida para apresentar também os resultados das outras simulações.

Os dados da tabela 7.1 mostram o quanto a falta de um mecanismo de balanceamento dinâmico no transporte afeta o desempenho do modelo. Como o parti-

Tabela 7.1: 10000 ciclos sem balanceamento dinâmico com REF=2 e particionamento por RCB

Número de processos	Tempo total	Tempo total da hidrodinâmica	Tempo do Solver da Hidrodinâmica	Tempo total do transporte	Tempo do solver do transporte
1	6794	1847	1146	4659	4192
2	5306(1.28)	1016(1.81)	631(1.81)	4050	247
			638(1.79)		3679
3	13155	8925	8657	4005(1.16)	318
			425(2.69)		34
			439(2.61)		3639
4	4798(1.41)	533(3.46)	308(3.72)	4047(1.15)	377
			329(3.48)		0.01
			323(3.55)		0.01
			331(3.46)		3674
5	2775(2.44)	444(4.15)	258	2197(2.12)	369
			270(4.71)		0.01
			262(4.37)		0.01
			278(4.12)		1660
			278(4.12)		1984
6	3681(1.84)	375(4.92)	210	3130(1.48)	347
			232		0.01
			225		0.01
			217		0.01
			229		822
			225		2837
7	3432(1.97)	343(5.38)	186	2934(1.58)	496
			204		0.01
			198		0.01
			202		0.01
			201		0.02
			213		2654
			202		1005

cionamento do domínio é feito baseado unicamente na hidrodinâmica, o ganho de desempenho no cálculo da hidrodinâmica é considerável, como pode ser verificado na coluna do tempo total da hidrodinâmica. O tempo total da hidrodinâmica para 7 processadores, por exemplo, é 5.38 vezes menor que o tempo para um único processador. É mostrado um único tempo total da hidrodinâmica para todos os processos porque, como ocorrem sincronizações entre os diversos processos a cada ciclo, o tempo total (processamento, espera na sincronização e comunicação) em que o processo fica no ciclo da hidrodinâmica é o mesmo para todos. Não foram feitas medidas mais detalhadas para identificar onde estão ocorrendo as perdas que reduzem a eficiência da paralelização, mas uma das fontes de perda são as comunicações entre os processos que ocorrem durante cada ciclo e entre os ciclos, para troca de dados nas fronteiras. Outra fonte de redução da eficiência é o uso de sobreposição na decomposição do domínio, que faz com que a soma dos subdomínios processados pe-

los processadores seja maior que o domínio total, acrescentando um processamento adicional.

A quarta coluna da tabela 7.1 mostra o tempo utilizado, em cada processador, na montagem das matrizes de coeficientes e na resolução dos sistemas resultantes, o que corresponde ao tempo total da hidrodinâmica em cada processador, descontado o tempo de comunicação e sincronização. Apesar de o algoritmo de particionamento (RCB) utilizado resultar em uma diferença de no máximo uma célula entre os subdomínios, a região de sobreposição necessária para o uso da decomposição de domínio com sobreposição depende da geometria do subdomínio e do tamanho das fronteiras entre o subdomínio e os subdomínios vizinhos, podendo ocorrer diferenças no número de células das regiões de sobreposição dos diferentes subdomínios. Essa diferença, entretanto, nos testes realizados não passou de 10%. Além do aumento dos subdomínios com a sobreposição, há também um aumento no número de matrizes montadas e sistemas a resolver quando é usado mais de um processador devido ao número de iterações na decomposição de domínio, até que as soluções locais convirjam para a solução global. Esses dois fatores reduzem a eficiência da decomposição de domínio.

A coluna 5 mostra o tempo total em que os processos ficaram no ciclo de transporte. Da mesma forma como ocorre na hidrodinâmica, como há sincronizações entre os processos a cada ciclo de transporte, o tempo total (processamento, espera na sincronização e comunicação) de todos os processos é o mesmo. A coluna 6 mostra individualmente, para cada processo, o tempo levado na montagem e solução dos sistemas resultantes do ciclo de transporte. Como o particionamento é feito baseado unicamente na hidrodinâmica, a distribuição da malha refinada de transporte de substâncias é bastante irregular do ponto de vista de balanceamento de carga, havendo processadores que não recebem nenhuma parte da malha. Há um ganho de desempenho com o aumento do número do processadores, mas esse ganho é pequeno em relação ao número de processadores e bastante irregular, havendo situações em que a alteração no particionamento da malha da hidrodinâmica, ao aumentar o número de processadores, piora o particionamento da malha do transporte, aumentando o tempo de processamento do transporte. É evidente, dos dados mostrados nessa tabela, a necessidade de um mecanismo de balanceamento de carga da malha refinada do transporte. Os dados da tabela mostram também que, para o domínio de hidrodinâmica e transporte utilizado nas medidas de desempenho, o cálculo do tranporte de substâncias ocupa uma parcela significativa do tempo total de processamento, justificando ainda mais a busca de mecanismos de balanceamento dinâmico para o transporte.

7.2 Modelo com as malhas desacopladas

A tabela 7.2 apresenta os resultados para o Guaíba 200 refinado com um refinamento igual a 2 (REF=2) resultando em um domínio com $\Delta X = \Delta Y = 100$ metros. Foi utilizada a abordagem das malhas desacopladas para o balanceamento dinâmico e, para o particionamento, o algoritmo RCB.

Os dados da tabela 7.2 mostram que a abordagem de efetuar o balanceamento da malha do transporte independentemente da malha da hidrodinâmica e buscar, a cada ciclo do transporte, os dados necessários para a interpolação, apresentou bons ganhos de desempenho. Os ganhos de desempenho no ciclo da hidrodinâmica são

Tabela 7.2: 10000 ciclos com malhas desacopladas com REF=2 e particionamento utilizando RCB

Número de processos	Tempo total	Tempo total da hidrodinâmica	Tempo do Solver da Hidrodinâmica	Tempo total do transporte	Tempo do solver do transporte
1	8245	2671	1860	5569	4633
2	4328 (1.9)	1436	1008 1009	2883 (1.9)	2312(0) 2181(1)
3	2966 (2.7)	978	669 688	1979 (2.8)	1534(0) 1393(2)
4	2357 (3.4)	744	496 523	1605 (3.4)	1202(0) 1040(3)
5	2273 (3.6)	938	708 756	1329 (4.2)	987(0) 821(4)
6	1731 (4.7)	519	335 368	1205 (4.6)	870(0) 678(5)
7	1809 (4.5)	711	513 573	1091 (5.1)	786(0) 562(6)
8	1458 (5.6)	397	248 278	1054 (5.3)	742(0) 416(7)
9	1526 (5.4)	534	366 538	984 (5.6)	694(0) 379(8)
12	1221 (6.7)	298	173 200	915 (6.0)	648(0) 216(9)
13	1299 (6.3)	448	312 355	844 (6.6)	580(0) 139(10)
14	1092 (7.5)	257	148 173	829 (6.7)	590(0) 135(10)
15	1265 (6.5)	397	266 309	860 (6.5)	590(0) 135(10)
16	1090 (7.5)	229	130 153	854 (6.5)	591(0) 136(10)
17	1205 (6.8)	358	233 283	841 (6.6)	591(0) 136(10)
18	1190 (6.9)	322	204 251	861 (6.5)	590(0) 136(10)
19	1087 (7.5)	255	156 190	825 (6.7)	590(0) 135(10)
20	1022 (8.0)	190	102 129	825 (6.7)	590(0) 136(10)

os mesmos obtidos na abordagem anterior, uma vez que o balanceamento dinâmico da malha refinada não afeta a malha não refinada e o cálculo da hidrodinâmica. Ao mesmo tempo em que foi mantido o ganho de desempenho da hidrodinâmica, obteve-se um ganho semelhante no cálculo do transporte.

Assim como na tabela anterior, a coluna 5 mostra o tempo total em que os processos ficaram no ciclo de transporte e a coluna 6 mostra, para o processo 0 e para o último processo onde ocorre transporte, o tempo levado na montagem e solução dos sistemas resultantes do ciclo de transporte. A diferença entre o valor da coluna 5 e cada um dos valores da coluna 6 mostra o tempo em que cada processo ficou efetuando comunicação, ficou em espera para sincronizar com os outros processos ou ficou calculando para todas as células do domínio algumas variáveis que são utilizadas na montagem da matriz de coeficientes. Verifica-se que para o processo 0 esse tempo é de aproximadamente 40% do tempo utilizado na montagem e solução das matrizes do ciclo de transporte. Pode-se observar também como o tempo utilizado na montagem das matrizes reduz-se à medida em que o processo está mais afastado do processo 0, onde inicialmente concentra-se toda a malha refinada. Esse resultado ocorre devido ao tempo necessário para que o mecanismo de difusão, utilizado para o balanceamento dinâmico, chegue a uma situação estável da distribuição da carga. Como o tempo representado na tabela inclui também os primeiros ciclos, quando a carga está concentrada no processo 0, o resultado final é uma diferença grande de carga entre o processo 0 e os demais processos, diferença essa que é função dos diversos parâmetros do mecanismo de balanceamento dinâmico, discutidos na seção 7.7.

7.3 Modelo com as malhas acopladas

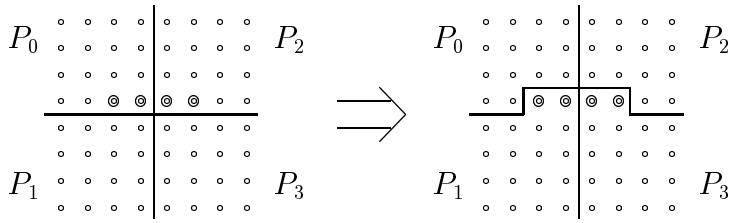
A tabela 7.3 apresenta os resultados para o Guaíba 200 sem refinamento, resultando em um domínio com $\Delta X = \Delta Y = 200$ metros. Foi utilizada a abordagem das malhas acopladas para o balanceamento dinâmico e, para o particionamento, o algoritmo RCB. Nas colunas 4 e 6 da tabela foram colocados dois valores por linha para redução do tamanho da mesma.

Tabela 7.3: 10000 ciclos com malhas acopladas com REF=1 e particionamento utilizando RCB

Número de processos	Tempo total	Tempo total da hidrodinâmica	Tempo do Solver da Hidrodinâmica		Tempo total do transporte	Tempo do solver do transporte	
1	2765	607	414		2157	1799	
2	1528	358	247	247	1168	954	835
3	1117	286	203	200	829	671	590
			205				513
4							
5	791	218	140	160	572	457	408
			148	162		346	291
			153			238	
6	738	196	125	145	541	434	373
			130	125		310	255
			131	120		204	152
7	687	158	88	111	527	421	360
			96	95		296	242
			95	103		191	136
			87			81	
8	655	128	74	86	526	421	360
			89	88		295	240
			81	86		186	128
			87	76		74	22
9	649	121	70	77	527	421	360
			85	81		295	240
			78	71		186	128
			77	74		74	22
			68			0	
10	643	117	59	67	525	419	361
			73	70		296	240
			70	68		186	128
			71	65		74	23
			68	61		0	0

Essa versão, apesar de apresentar *speedups* razoáveis, mesmo para um subdomínio pequeno (o Guaiba 200, sem refinamento, possui 11506 células na malha de hidrodinâmica e a região refinada utilizada nos testes inicia com 51 x 63 células, totalizando 3213 células), ocasionou um erro de execução ao executar 10000 ciclos com 4 processos ou mais de 10 processos. A depuração do modelo mostrou que em algumas situações, quando ocorriam transferências simultâneas de carga entre as malhas

da hidrodinâmica de processos vizinhos, ao final das trocas de mensagens necessárias para as transferências, as descrições do estêncil e da região de sobreposição estavam incorretos. Essa situação, ilustrada na figura 7.1, é descrita a seguir.



(a) Situação inicial (b) Situação final

Figura 7.1: Transferência simultânea de células entre subdomínios vizinhos

Na situação mostrada na figura 7.1(a), o processo P_0 deve enviar para o processo P_1 as duas células marcadas no subdomínio de P_0, assim como P_2 deve enviar para P_3 as células marcadas em seu subdomínio, resultando na situação mostrada na figura 7.1(b). Entretanto, as células que serão transferidas de P_2 para P_3 fazem parte da sobreposição de P_2 em P_0 e essa informação deve ser passada por P_0 para P_1, para que P_1 possa solicitar de P_2 as células atualizadas a cada ciclo. Assim, após a transferência P_1 tem a informação de que as células da sobreposição estão ainda em P_2, quando na verdade as mesmas já foram transferidas para P_3, causando uma inconsistência na descrição da sobreposição. Essa situação pode ocorrer de diversas formas envolvendo o estêncil e sobreposição e pode ser resolvida mediante trocas adicionais entre os processos até que as descrições dos estênceis e sobreposições sejam consistentes.

A solução adotada até o momento para resolver o problema citado foi restringir as transferências entre processos vizinhos, de forma que apenas um processo a cada instante pode estar enviando carga para outro processo. Como esse problema ocorreu apenas no particionamento irregular da malha da hidrodinâmica onde os desbalanceamentos são bem menores, essa restrição não afetou o desempenho do modelo, pelo menos até o número de processos utilizado nos testes. Após colocada essa restrição, foram efetuados novos testes no modelo. A tabela 7.4 apresenta os resultados para o mesmo domínio, o Guaíba 200 sem refinamento, com a restrição de dois domínios não poderem enviar células da malha da hidrodinâmica para outros domínios no mesmo ciclo. Nessa tabela são apresentados o tempo mínimo do *solver* da hidrodinâmica e o tempo máximo do *solver* da hidrodinâmica, ou seja, o tempo levado na montagem e solução dos sistemas da hidrodinâmica pelo processo que levou menos tempo e pelo processo que levou mais tempo.

Pode-se observar na coluna 6 da tabela que os tempos utilizados por cada processador para resolução da malha refinada do transporte não variam a partir de 8 processos. Esse comportamento é devido ao pequeno número de células da malha refinada. Após ser alcançada a situação mostrada na figura 7.2, onde a carga está bem distribuída e os subdomínios são pequenos, o acréscimo de mais processadores não faz com que partes da malha refinada sejam enviadas aos mesmos porque a quantidade de carga a ser enviada e os tamanhos dos subdomínios resultantes dessa divisão seriam menores do que o mínimo definido para o modelo.

Pode-se observar também a diferença no tempo de montagem e solução das matrizes da hidrodinâmica entre os processos. Em alguns particionamentos a diferença

Tabela 7.4: 10000 ciclos com malhas acopladas com REF=1 e particionamento utilizando RCB sem transferências simultâneas de carga na hidrodinâmica

Número de processos	Tempo total	Tempo total da hidrodinâmica	Tempo do Solver da Hidrodinâmica (min/max)	Tempo total do transporte	Tempo do solver do transporte
1	2761	605	411	2155	1796
2	1567 (1.7)	398	286 279	1168	1567(0) 830(1)
3	1141 (2.4)	310	219 229	830	671(0) 509(2)
4	895 (3.0)	234	145 169	659	529(0) 345(3)
5	796 (3.4)	223	135 166	571	456(0) 237(4)
6	730 (3.7)	187	108 137	542	434(0) 151(5)
7	689 (4.0)	161	82 114	526	421(0) 81(6)
8	659 (4.2)	131	69 90	526	420(0) 22(7)
9	649 (4.3)	120	58 84	526	420(0) 22(7)
10	639 (4.3)	109	54 74	528	420(0) 22(7)
11	636 (4.3)	105	52 72	529	420(0) 22(7)
12	627 (4.4)	98	48 65	528	420(0) 22(7)
13	735 (3.7)	204	44 139	530	420(0) 22(7)
14	759 (3.6)	227	41 175	532	420(0) 22(7)
15	739 (3.7)	207	37 152	530	420(0) 22(7)
16	733 (3.7)	200	35 144	531	420(0) 22(7)
17	738 (3.7)	205	36 138	531	420(0) 22(7)
18	709 (3.9)	176	35 136	531	420(0) 22(7)
19	689 (4.0)	156	33 121	531	420(0) 22(7)
20	684 (4.0)	150	31 110	532	420(0) 22(7)

entre o processo que levou mais tempo e o processo que levou menos tempo é pequena, enquanto que em outros particionamentos essa diferença é bem maior. O particionador utilizado (RCB) garante que a diferença de células entre dois subdomínios é no máximo de uma célula, mas as geometrias diferentes dos subdomínios fazem com que o tamanho das áreas de sobreposição difiram bastante, o que pode causar essas diferenças nos tempos de execução da hidrodinâmica.

Para avaliar a escalabilidade do método implementado à medida em que o número de células do domínio crescia foram feitos testes com o Guaíba 200 com um refinamento igual a 2 e 3. A tabela 7.5 apresenta os resultados para o Guaíba 200 refinado com um refinamento igual a 2 (REF=2) resultando em um domínio com $\Delta X = \Delta Y = 100$ metros e a tabela 7.6 apresenta os resultados para o Guaíba 200 refinado com um refinamento igual a 3 (REF=3) resultando em um domínio com $\Delta X = \Delta Y = 66$ metros. Em ambas execuções foi utilizada a abordagem das malhas acopladas para o balanceamento dinâmico, com a restrição de dois processos ou mais não poderem enviar carga da malha de hidrodinâmica no mesmo ciclo. Para o particionamento estático da hidrodinâmica foi utilizado o algoritmo RCB.

Pode-se observar na tabela 7.6 que processos a partir do 15 (inclusive) não recebem malha do transporte para processar de forma que o acréscimo de mais processos não reduz o tempo do transporte. Os dados mostrados não permitem concluir se isso ocorreu porque os 10.000 ciclos do teste não foram suficientes para que o mecanismo

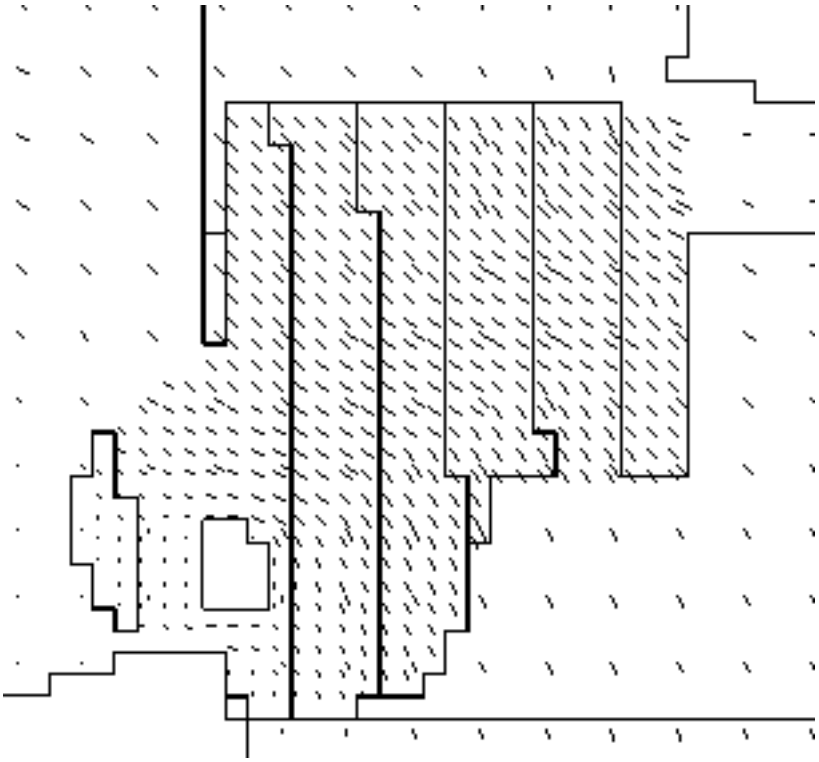


Figura 7.2: Malha refinada após 10000 ciclos

de difusão levasse carga até os processos a partir do 15 ou se o tamanho da malha refinada não é suficiente para que os processos a partir do 15 recebam carga, uma vez que abaixo de um determinado tamanho os subdomínios não são mais particionados. Para obter mais informações sobre esse ponto foi feito um novo teste com REF=3, executando 15.000 ciclos. Os resultados desse teste encontram-se na tabela 7.7.

O teste mostra que ao aumentar o número de ciclos de 10.000 para 15.000, além dos processos de 0 a 14 que já recebiam carga de transporte nos 10.000 ciclos, apenas o processo 15 passou a receber carga de transporte. Isso pode ter ocorrido devido à malha refinada ter tido um crescimento pequeno mas também à distância entre os processos 0 e 15, que faz com que, no particionamento por faixas utilizado, alterações na carga dos processos de menor ordem levem um número grande de ciclos até afetar os processos de maior ordem. Para eliminar da avaliação da convergência do algoritmo o crescimento da malha, na seção 7.7 são feitos diversos testes da convergência utilizando uma malha refinada estática.

7.4 Comparação das duas abordagens

A comparação entre a abordagem das malhas desacopladas e a abordagem das malhas acopladas foi feita baseando-se nos testes de 10.000 ciclos com refinamento igual a 2 e particionamento por RCB (tabelas 7.2 e 7.5).

Ao iniciar este trabalho esperava-se que a versão de malhas acopladas apresentaria um desempenho melhor que a versão com as malhas desacopladas, uma vez que não há o custo adicional da transferência das células da hidrodinâmica que são necessárias para o transporte e residem em outro processador. Entretanto, ambas

Tabela 7.5: 10000 ciclos com malhas acopladas com REF=2 e particionamento utilizando RCB

Número de processos	Tempo total	Tempo total da hidrodinâmica	Tempo do Solver da Hidrodinâmica		Tempo total do transporte	Tempo do solver do transporte	
1	8652	2692	1852		5955	4995	
2	4758 (1.81)	1629	1110	1164	3124	2580	2418(1)
3	3361 (2.57)	1178	802	855	2178	1775	1589(2)
4	2666 (3.24)	958	624	697	1704	1375	1170(3)
5	2243 (3.85)	815	563	602	1423	1143	915(4)
6	2044 (4.23)	802	489	600	1236	991	736(5)
7	2781 (3.11)	1665	402	1498	1110	887	602(6)
8	1608 (5.38)	577	370	431	1025	818	493(7)
9	1647 (5.25)	669	319	516	972	772	388(8)
10	1551 (5.57)	611	266	485	935	743	306(9)
11	1354 (6.39)	431	266	309	917	728	237(10)
12	1300 (6.65)	386	233	260	908	721	159(11)
13	1278 (6.77)	363	195	243	909	720	84(12)
14	1255 (6.89)	340	185	225	910	721	17(13)
15	1238 (6.98)	321	150	216	911	721	17(13)
16	1219 (7.09)	303	139	199	910	721	17(13)
17	1203 (7.19)	286	138	192	911	647	17(13)
18	1190 (7.27)	273	139	181	912	722	17(13)
19	1180 (7.33)	264	114	175	911	721	17(13)
20	1166 (7.42)	250	119	165	911	721	17(13)

abordagens apresentaram *speedups* bastante aproximados no tempo total, com vantagem ora para uma, ora para a outra abordagem. Os tempos de processamento da malha refinada foram praticamente os mesmos em ambas abordagens, ocorrendo as diferenças no processamento da malha da hidrodinâmica. Essa diferença no *speedup* no processamento da hidrodinâmica pode ser explicado pelo aumento das fronteiras que ocorre na malha da hidrodinâmica quando é utilizada a abordagem das malhas acopladas, aumento de fronteiras esse que não ocorre na abordagem das malhas desacopladas, uma vez que o particionamento da malha refinada não afeta o particionamento da malha da hidrodinâmica. Os resultados obtidos mostram que o custo da transferência das células da hidrodinâmica que ocorre a cada ciclo na abordagem das malhas desacopladas equivale aproximadamente ao custo da transferência das células nas fronteiras maiores e ao processamento adicional devido à maior área de sobreposição que resulta das fronteiras maiores.

7.5 Efeito do particionador utilizado

Para comparar o efeito das partições geradas pelos particionadores no desempenho do modelo foram efetuadas medidas sobre o mesmo domínio utilizando como particionador o pacote METIS e o algoritmo RCB. Foram executados 10.000 ciclos apenas de hidrodinâmica, sem malha refinada, para o mesmo domínio do teste anterior (Guaíba 200 refinado com um refinamento igual a 2 resultando em um domínio

Tabela 7.6: 10000 ciclos com malhas acopladas com REF=3 e particionamento utilizando RCB

Número de processos	Tempo total	Tempo total da hidrodinâmica	Tempo do Solver da Hidrodinâmica	Tempo total do transporte	Tempo do solver do transporte
1	16519	5474	3718	11036	9297
2	9004	3261	2307 2184	5733	
3	6330	2303	1542 1654	4017	
4	5021	1807	1179 1301	3205	
5	4332	1593	1112 1180	2729	
6	3814	1360	869 1002	2443	
7	3431	1163	771 846	2257	
8	3235	1090	706 809	2135	
9	3014	947	607 691	2056	
10	2806	795	454 562	2000	
11	2789	812	501 570	1967	
12	2675	711	401 479	1953	
13	2628	678	362 458	1938	
14	2580	625	344 417	1944	
15	2551	599	277 397	1941	1566 1324 1140 983 852 735 630 536 445 367 298 223 145 59 0.01
16	2521	568	310 374	1943	
17	2506	551	265 356	1944	
18	2476	515	261 339	1944	
19	2441	486	212 316	1944	
20	2419	464	229 300	1944	

com $\Delta X = \Delta Y = 100$). A tabela 7.8 apresenta os tempos de execução para ambos particionadores, para número de processadores variando entre 1 e 20.

Para poder avaliar a qualidade das partições geradas, a tabela 7.9 apresenta o tempo total da execução de 10.000 ciclos da hidrodinâmica sendo utilizado o algoritmo RCB, bem como o número de células internas de cada subdomínio, o número de células da região de sobreposição para o subdomínio com menor área de sobreposição e para o subdomínio com maior área de sobreposição, o número de arestas externas do subdomínio com menor número de arestas externas e do subdomínio com maior número de arestas externas (sendo mostrado em parênteses o número de subdomínios vizinhos de cada um) e o número mínimo de subdomínios vizinhos e o número máximo de vizinhos para a partição. Esses dados são apresentados para um número de processos variando de 1 a 20.

A tabela 7.10 apresenta os mesmos dados da tabela 7.9 para uma execução de 10.000 ciclos da hidrodinâmica sendo utilizada a biblioteca MÉTIS. Conforme reco-

Tabela 7.7: 15000 ciclos com malhas acopladas com REF=3 e particionamento utilizando RCB

Número de processos	Tempo total	Tempo total da hidrodinâmica	Tempo do Solver da Hidrodinâmica	Tempo total do transporte	Tempo do solver do transporte
1	25686	8236	5579	17440	14664
2	14002	4921	3477 3296	9071	7547 7079
3	9821	3482		6329	5225 4752 4629
4	7746	2758		4979	
5	6632	2421		4202	
6	5752	2027		3714	
7	5167	1773		3384	
8	4801	1645		3146	
9	4454	1455		2988	
10	4133	1237		2866	
11	4048	1250		2786	
12	3836	1089		2737	
13	3748	1038		2699	
14	5058	1774		3273	
15	3655	961		2683	2163 1861 1641 1451 1284 1137 1011 886 767 663 575 474 373 261 42
16	3586	892		2683	
17	3561	865		2685	
18	3490	793		2686	
19	3441	746		2684	
20	3402	707		2684	

mendado na documentação do MÉTIS, para um particionamento até 7 subdomínios foi utilizada a rotina *METIS_PartGraphRecursive*, que executa uma distribuição de carga com uma diferença máxima de uma célula entre subdomínios vizinhos. Para mais de 7 subdomínios foi utilizada a rotina *METIS_PartGraphKway* que permite uma diferença entre cargas de até 10%. Na tabela são mostrados o tamanho mínimo e o tamanho máximo dos subdomínios gerados. Conforme descrito na seção 2.3.1, o MÉTIS utiliza para particionamento um algoritmo multinível, no qual o grafo é inicialmente reduzido, através da condensação de vértices, a um grafo menor, sobre o qual é aplicado um algoritmo de particionamento. Após o particionamento o grafo é novamente expandido ao grafo original, sendo que nessa expansão é utilizado um algoritmo de melhoramento como o Fiduccia-Mattheyses. O MÉTIS permite selecionar, em ambas as funções de particionamento, qual a heurística utilizada na contração das arestas, podendo ser escolhida a cada passo a aresta de maior peso ou uma

Tabela 7.8: Tempo de execução do METIS x RCB

Número de processos	Tempo total com o METIS	Tempo total com o RCB	Número de processos	Tempo total com o METIS	Tempo total com o RCB
1	2430	2427	10	367	351
2	1350	1413	11	352	317
3	966	1017	12	311	303
4	704	737	13	304	286
5	597	629	14	272	263
6	529	516	15	261	243
7	455	455	16	253	234
8	403	395	17	236	227
9	383	361	18	225	217

Tabela 7.9: Resultados obtidos no particionamento com o RCB

	Tempo total	Número de Células Internas	Número de Células de Sobreposição (min/max)	Número de arestas (min/max)	Número de Subdomínios vizinhos (min/max)
1	2427	46024	0/0	0(0)/0(0)	0/0
2	1413	23011	407/415	203(1)/203(1)	1/1
3	1017	15342	318/589	155(1)/298(2)	1/2
4	737	11506	152/518	65(2)/260(3)	2/3
5	629	9205	202/579	95(2)/294(4)	1/4
6	516	7671	200/624	91(3)/309(5)	2/5
7	455	6574	152/629	67(3)/317(5)	2/5
8	395	5754	118/541	51(3)/274(4)	2/6
9	361	5114	122/586	51(3)/289(6)	2/6
10	351	4602	106/573	47(3)/286(5)	2/6
11	317	4185	138/561	61(2)/281(6)	2/6
12	303	3836	150/557	67(3)/277(6)	2/6
13	286	3541	146/504	65(2)/244(4)	2/7
14	263	3288	150/494	65(3)/243(5)	2/7
15	243	3069	150/480	67(2)/233(6)	2/7
16	234	2876	154/462	69(3)/226(6)	3/7
17	227	2707	154/500	69(2)/246(3)	2/8
18	217	2557	148/512	64(3)/250(6)	3/8
19		2423	140/460	59(2)/226(5)	2/8
20		2302	116/484	51(3)/237(6)	2/8
25		1843	142/372	61(3)/178(7)	3/7

aresta aleatória. Nos testes foi utilizada a seleção aleatória da aresta, uma vez que todas as arestas do grafo tem o mesmo peso. Na função *METIS_PartGraphRecursive* o particionamento do grafo reduzido é feito pelo algoritmo de crescimento de região(*Region Growing*[1]) enquanto que na função *METIS_PartGraphKway* o partici-

[1]No algoritmo de Crescimento de Região inicialmente seleciona-se um vértice aleatoriamente e

onamento do grafo reduzido é feito pelo algoritmo de bissecção recursiva multinível. Na função *METIS_PartGraphRecursive* o algoritmo de refinamento é uma variante do Fidducia-Mattheyses, enquanto que na função *METIS_PartGraphKway* pode-se escolher o algoritmo de refinamento tendo sido utilizado nos testes um algoritmo de seleção aleatória de arestas. Não foram executados testes com todas as opções oferecidas de contração, particionamento e refinamento por fugir um pouco ao escopo deste trabalho. Na tabela 7.10 é mostrado também, entre parênteses, ao lado do tempo total, o quanto a diferença em relação ao tempo total da execução com o RCB representa em percentual.

Tabela 7.10: Resultados obtidos no particionamento com o MÉTIS

	Tempo total	Número de Células Internas	Número de Células de Sobreposição (min/max)	Número de arestas (min/max)	Número de Subdomínios vizinhos (min/max)
1	2430	46024	0/0	0(0)/0(0)	0/0
2	1350 (-4.6)	23012	276/292	138(1)/138(1)	1/1
3	966 (-5.2)	15341	224/502	100(1)/246(2)	1/2
4	704 (-4.6)	11506	152/524	70(2)/256(2)	2/2
5	597 (-5.3)	9205	232/504	116(2)/246(2)	2/2
6	529 (+2.4)	7670	208/604	98(2)/294(3)	2/3
7	455 (0.0)	6575	148/508	68(2)/248(2)	2/3
8	403 (+1.9)	5644/5908	110/520	52(2)/256(3)	1/4
9	383 (+5.7)	4964/5256	116/575	52(2)/282(5)	1/5
10	367 (+4.3)	4482/4708	120/544	50(3)/264(4)	2/5
11	352 (+9.9)	4064/4305	210/604	103(1)/292(4)	1/5
12	311 (+2.5)	3734/3944	128/520	60(2)/252(4)	2/5
13	304 (+5.9)	3450/3625	144/568	60(3)/274(4)	1/7
15	261 (+3.3)	2933/3153	125/492	60(2)/240(6)	1/6
16	253 (+6.8)	2792/2957	116/544	54(2)/272(4)	2/5
18	225 (+3.5)	2482/2749	108/524	50(2)/252(6)	2/6

A comparação entre as tabelas mostra que o corte de arestas do subdomínio de maior número de arestas externas gerado pelo MÉTIS foi melhor em quase todas as partições resultando em uma área de sobreposição menor quando o domínio era particionado em poucos subdomínios. O corte de arestas menor e a área de sobreposição menor resultaram em um desempenho da hidrodinâmica do modelo aproximadamente 5 % melhor quando o número de subdomínios era pequeno (até 5 subdomínios). Quando o número de subdomínios crescia, apesar de o corte de arestas do MÉTIS ainda ser melhor na maior parte dos casos, o número de células de sobreposição nas partições geradas pelo RCB era menor, resultado da maior regularidade dos subdomínios. A menor área de sobreposição aliada ao melhor balanceamento da carga fizeram com que a partir de 6 processos o RCB apresentasse um desempenho aproximadamente 5 % melhor que o MÉTIS.

vai-se adicionando vértices segundo um critério de ganho semelhante ao utilizado nos algoritmos Kernighan-Lin e Fiduccia-Mattheyses até ser atingido o número de vértices buscados na partição (KARYPIS; KUMAR, 1998d).

7.6 Troca de Mensagens por memória compartilhada e contenção no acesso à memória compartilhada

Em um *cluster* com nodos multiprocessados como o utilizado nesse trabalho, a troca de mensagens entre processos residentes em processadores pertencentes ao mesmo nodo pode ser feita pela memória compartilhada, sendo que a opção por essa forma de comunicação é feita na instalação do MPICH. Para avaliar o desempenho obtido quando é usada memória compartilhada para comunicação entre subdomínios vizinhos, foram feitas execuções em três situações diferentes, com o domínio particionado em 2 subdomínios. Nesses testes foi utilizada uma versão do MPICH configurada para efetuar a troca de mensagens entre processos do mesmo nodo utilizando a memória compartilhada e uma versão configurada para não utilizar a memória compartilhada. Na primeira delas, foram utilizados dois processos no mesmo nodo, na versão configurada para não utilizar memória compartilhada. Na segunda situação, foram utilizados dois processos no mesmo nodo na versão configurada para utilizar memória compartilhada e no terceiro caso foram utilizados dois processos em nodos diferentes. Nos três testes foram executados 2000 ciclos sobre o domínio do teste anterior (Guaíba 200 refinado com um refinamento igual a 2 resultando em um domínio com $\Delta X = \Delta Y = 100$ metros), sendo utilizada a abordagem das malhas acopladas para o balanceamento dinâmico e o RCB para o particionamento estático da hidrodinâmica. A tabela 7.11 mostra os resultados obtidos nos três testes.

Tabela 7.11: Desempenho com troca de mensagens por memória compartilhada

	Tempo total	Tempo total da hidrodinâmica	Tempo do Solver da Hidrodinâmica		Tempo total do transporte	Tempo do solver do transporte	
2 processos em nodos diferentes	781	282	195	184	494	409	382
2 processos no mesmo nodo não comunicando-se por memória compartilhada	813	309	205	195	498	411	385
2 processos no mesmo nodo comunicando-se por memória compartilhada	819	311	207	197	503	416	390

A tabela mostra que quando os dois processos estão rodando no mesmo nodo, a diferença de desempenho entre o uso e o não uso da memória compartilhada para troca de mensagens é muito pequena, de menos de 1%, sendo que essa diferença é a favor da versão que não utiliza comunicação por memória compartilhada. Uma diferença de desempenho bem maior ocorreu quando os dois processos estão em nodos diferentes. Nessa situação o tempo de execução da versão com um processo em cada nodo foi aproximadamente 5 % menor. Um fator que poderia causar esse comportamento seria a concorrência pela CPU entre a aplicação sendo executada e *daemons* e outros processos do Sistema Operacional, quando a aplicação necessita

das duas CPUs do nodo por ter dois processos. Nesse caso, os processos do Sistema Operacional teriam que compartilhar as CPUs com a aplicação. Tal não ocorre quando a aplicação possui apenas um processo no nodo, utilizando por isso apenas uma CPU, ficando a outra CPU livre para rodar processos do Sistema Operacional. A avaliação do uso da CPU com o utilitário *top* do Linux, entretanto, mostrou que o uso da CPU por processos do Sistema Operacional é mínima, ficando abaixo do 1% da CPU.

Outro fator que poderia causar um pior desempenho quando os processos estão executando no mesmo nodo é a contenção no acesso à memória compartilhada pelos processadores no mesmo nodo. Para avaliar essa contenção foram feitos testes em que foram rodadas duas instâncias independentes do modelo no mesmo nodo e, após, foram rodadas as mesmas duas instâncias em nodos diferentes. A execução de duas instâncias independentes do modelo com um único domínio em cada instância ao invés de uma única instância com dois processos, dividindo o domínio entre os dois processos, faz com que não haja comunicação entre os dois processos, de forma que a diferença de desempenho detectada entre as execuções não está relacionada à forma como é feita a troca de mensagens. Sendo eliminados desse teste a influência da comunicação e de outros processos do Sistema Operacional, as diferenças entre as execuções devem ser resultado da contenção que ocorre no primeiro teste, onde os processos compartilham a memória, e não deve ocorrer no segundo, onde os processos estão em nodos diferentes. A tabela 7.12 mostra o resultado dos testes.

Tabela 7.12: Contenção de memória

	Tempo total
Apenas 1 processo em L1	699
2 processos independentes em L1	746
1 processo em L1 e 1 processo em L2	699

A tabela mostra que o tempo de execução de um processo aumenta de forma significativa (no teste realizado aumentou em 6.7%) quando há outro processo sendo executado no mesmo nodo. Uma vez que os testes anteriores mostraram que a concorrência de outros processos do Sistema Operacional concorrendo com a aplicação não afeta de forma significativa o desempenho da aplicação, pode-se atribuir à contenção no acesso à memória compartilhada essa queda de desempenho. A realização de mais testes com diferentes refinamentos do modelo, utilizando memória de maneira mais intensiva poderia mostrar se há relação entre a quantidade de memória utilizada e a contenção bem como estudar o comportamento dessa relação. Poderia ocorrer também, em modelos com menor refinamento, que a maior parte dos dados permanecesse na memória *cache*, tornando insignificante a perda por contenção. Esses estudos, entretanto, não são o objetivo desse trabalho, sendo que os testes realizados nessa seção visavam apenas fornecer alguns dados para uma discussão sobre o uso de máquinas multiprocessadas.

Uma outra situação que pode ocorrer, se os dois processadores tiverem uma memória *cache* compartilhada, é que a execução dos dois processos faça com que a taxa de acertos da *cache* caia afetando o desempenho do modelo. Medidas da taxa de acerto podem ser obtidas com um perfilador de código como o *valgrind* ou obtendo esses dados diretamente das CPUs pela aplicação.

Os resultados mostrados nessa seção apontam que, pelo menos em algumas si-

tuações, *clusters* formados por máquinas biprocessadas podem apresentar um desempenho inferior a *clusters* formados por máquinas monoprocessadas com igual número de processadores. É possível que, se ambos processadores tiverem uma memória *cache* independente e suficientemente grande, a contenção seja reduzida fazendo com que o ganho de comunicação entre os processos no mesmo nodo em relação à comunicação entre os processos de nodos diferentes torne vantajoso o uso de máquinas biprocessadas. Para avaliar essas hipóteses, entretanto, são necessários testes que fogem ao escopo deste trabalho.

7.7 Avaliação da convergência do algoritmo

Para avaliar o efeito dos diversos parâmetros na convergência do algoritmo de difusão utilizado, foram efetuados testes com uma malha refinada composta por um total de 27.342 células. Foram utilizados 10 processos e mediu-se o número de ciclos necessários para que a diferença de carga entre dois processos vizinhos ficasse abaixo de um determinado valor, diferente para cada teste. Inicialmente foram feitos testes com 10000 ciclos, utilizando uma diferença máxima de carga igual a 200 células e com o percentual da diferença da carga a ser transferido variando entre 40% e 70%. A tabela 7.13 mostra o número de células em cada um dos 10 processos ao ser atingida uma distribuição de carga estável, para cada percentual da diferença de carga a ser transferida, e ao lado do percentual, entre parênteses, se encontra o número de ciclos necessário para atingir essa distribuição.

Tabela 7.13: Distribuição final da carga para 10 processadores e diferença máxima de carga igual a 200 células

	40%(4420)	50%(3170)	60%(2170)	70%(1420)
1	3447	3402	3303	3150
2	3285	3258	3159	2988
3	3123	3087	2997	2862
4	2943	2907	2844	2745
5	2772	2745	2691	2925
6	2709	2601	2772	2790
7	2529	2448	2628	2709
8	2349	2448	2475	2556
9	2178	2295	2304	2394
10	2007	2151	2169	2223

A tabela 7.13 mostra que o valor adotado para a diferença de carga tolerada entre dois subdomínios vizinhos antes de ser feita a transferência, igual a 200 células, resulta em uma diferença significativa de carga entre o primeiro e o último processos no particionamento por faixas adotado. A distribuição de carga estabiliza numa situação em que a carga do primeiro processo chega a ser 70% maior do que a carga do último processo, e esse desbalanceamento se mantêm por todo o resto da simulação, mostrando que as diferença de carga verificadas nos testes anteriores não são devidas apenas ao período em que a carga ainda não estava balanceada, mas também devidas a essa diferença que se mantem durante toda a simulação.

O teste mostra também o efeito do percentual da diferença de carga a ser transferido na convergência do algoritmo de difusão. Para um percentual de 40% foram

necessários 4420 ciclos até atingir uma distribuição estável de carga enquanto que para um percentual de 70% foram necessários apenas 1420 ciclos sugerindo que quanto mais alto o percentual da diferença a ser transferido, mais rápida a convergência. Sabe-se, entretanto, que percentuais altos na difusão podem levar à ocorrência de oscilações na carga, onde parte da carga fica sendo continuamente transferida entre dois processadores, acarretando custo de comunicação. As figuras 7.3 a 7.6 mostram o comportamento da distribuição da carga até atingir a situação de equilíbrio, para diferentes valores de percentual da diferença a ser transferido.

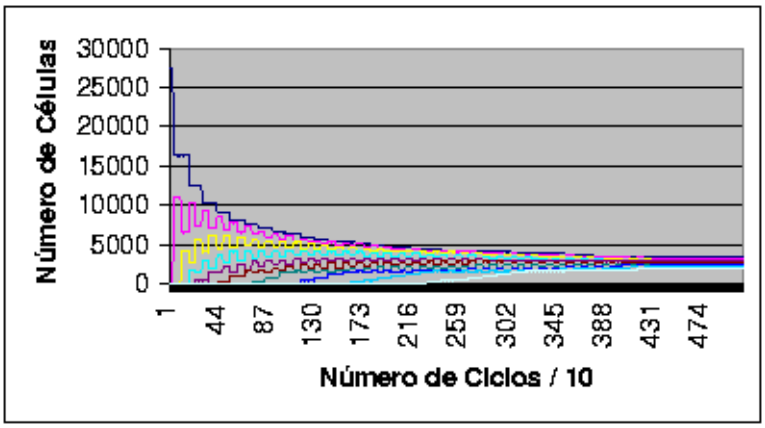


Figura 7.3: Comportamentco da carga com PERC_DIF=40

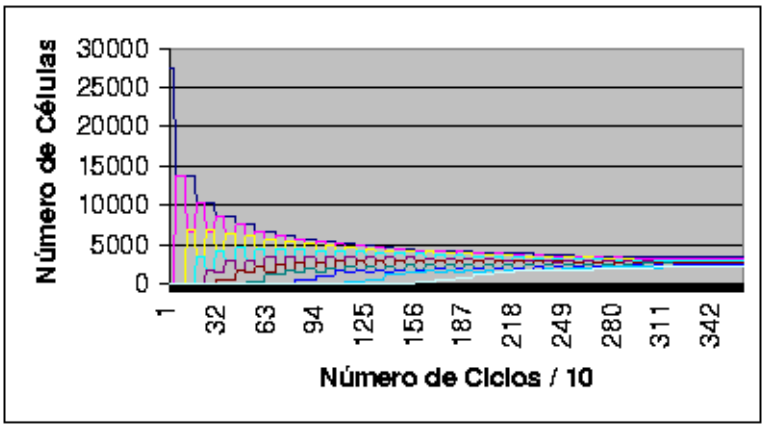


Figura 7.4: Comportamento da carga com PERC_DIF=50

As figuras mostram as oscilações na carga de cada processador até ser atingida uma distribuição estável, sendo que a amplitude de cada oscilação corresponde à carga que foi recebida ou transferida naquele ciclo. Assim, uma convergência em menos ciclos mas com oscilações de maior amplitude pode ter um custo maior que uma convergência mais lenta (em número de ciclos) mas de amplitude menor. Para verificar se tal ocorre foram efetuados outros testes, com 10 processadores, com uma diferença mínima de carga igual a 20 células, e com percentuais de diferença de carga a serem transferidos variando entre 40% e 80% e foi medido não apenas o número de ciclos até a convergência, mas o tempo total utilizado para os 10.000 ciclos. Os resultados desses testes se encontram nas tabelas 7.14 e 7.15.

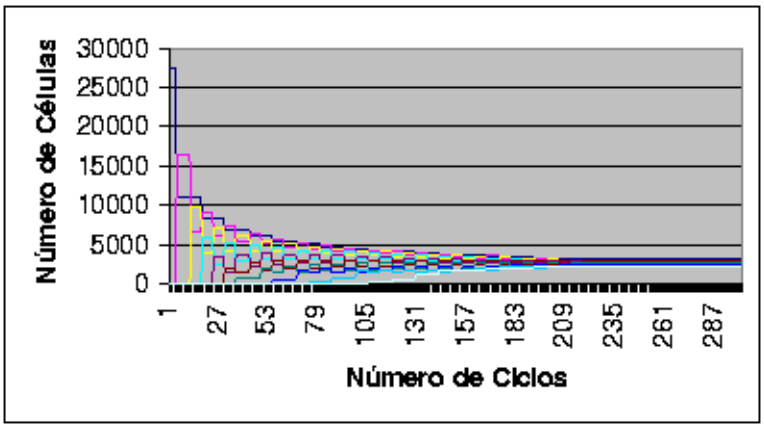


Figura 7.5: Comportamento da carga com PERC_DIF=60

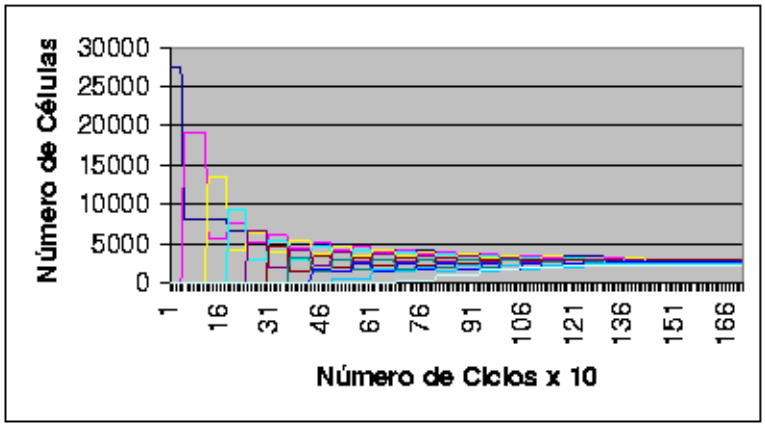


Figura 7.6: Comportamento da carga com PERC_DIF=70

Tabela 7.14: Número de células em cada processador após 10000 ciclos e diferença máxima de carga igual a 20 células

	Percentuais da diferença de carga a ser transferida				
	40%	50%	60%	70%	80%
1	2790	2808	2772	2754	2627
2	2772	2790	2781	2772	2627
3	2754	2772	2772	2754	2709
4	2745	2754	2754	2736	2703
5	2736	2736	2736	2718	2781
6	2745	2718	2718	2754	2772
7	2727	2718	2727	2736	2799
8	2709	2700	2709	2718	2790
9	2691	2682	2691	2709	2829
10	2673	2664	2682	2691	3078

A tabela 7.14 mostra o número de células em cada um dos 10 processos uma vez atingida uma distribuição estável de carga, na qual a diferença de carga entre dois subdomínios vizinhos não ultrapassasse 20 células. Não há diferenças significativas

Tabela 7.15: Número de ciclos até a convergência e tempo total para 10000 ciclos com diferença máxima de carga igual a 20 células

	Percentuais da diferença de carga a ser transferida			
	40%	50%	60%	70%
Nro. de Ciclos	8530	5540	3790	2420
Tempo total	1134	1102	1084	1087

entre as distribuições alcançadas com exceção da situação onde a carga transferida é igual a 80% da diferença de carga entre os processos vizinhos. Nessa situação o mecanismo de balanceamento não alcançou uma configuração de carga dentro da diferença máxima permitida. A tabela 7.15 mostra o número de ciclos necessários para atingir a distribuição de carga balanceada e o tempo total para a execução do transporte com para cada um dos percentuais avaliados. Os testes mostraram que o número de ciclos para a convergência a uma situação de carga balanceada reduz-se significativamente com o aumento do percentual da diferença de carga a ser transferida, até um percentual próximo a 70%. A partir desse percentual o esquema oscila e não converge para uma situação de carga bem distribuída.

Para avaliar o efeito dos parâmetros de diferença máxima de carga e percentual da diferença a ser transferida, um dos testes da seção 7.3 foi refeito com a diferença máxima de carga igual a 15 células e com o percentual da diferença a ser transferido igual a 50%. Foram rodados 10000 ciclos com refinamento igual a 3 e foi usado como particionador o RCB. A tabela 7.16 mostra o resultado desta execução. A comparação das tabelas 7.16 e 7.6 mostra que como resultado da menor diferença de carga tolerada entre os subdomínios vizinhos e do percentual maior de diferença a ser transferido, a malha refinada foi distribuída por um número maior de processos, chegando até o processo 15 e o tempo de execução do transporte foi reduzido, quando utilizados 20 processos, de 1944 segundos para 1642 segundos, reduzindo assim o tempo total da simulação..

7.8 Uso do modelo em ambientes compartilhados

Para avaliar a usabilidade dos mecanismos aqui apresentados em ambientes compartilhados por outras aplicações, foi implementada uma versão do balanceador que utiliza como medida de avaliação de carga o tempo de processamento do subdomínio, conforme discutido na seção 6.4.

No mecanismo implementado, cada processo P_i troca com os processos P_{i+1} e P_{i-1}, responsáveis pelos subdomínios vizinhos, informações sobre o número de células e o tempo de montagem e solução das matrizes e se a diferença de tempo estiver acima de um determinado valor percentual, o processo que leva mais tempo para processar seu subdomínio envia uma parte do mesmo para o processo vizinho que leva menos tempo para processar seu subdomínio. O número de células a ser enviado do processo P_i para o processo P_{i+1} ou P_{i-1} é calculado a partir da capacidade de processamento disponível de cada processador envolvido na transferência e do número de células de cada um. A capacidade de processamento disponível do processador P_i, representada por C_i, é dada por $C_i = N_i/T_i$, onde N_i é o número de células do subdomínio tratado pelo processo P_i e T_i é o tempo, em segundos, levado

Tabela 7.16: 10000 ciclos com malhas acopladas com REF=3 e particionamento utilizando RCB com Perc_Dif=50

Número de processos	Tempo total	Tempo total da hidrodinâmica	Tempo total do transporte	Tempo do solver do transporte	
1	16647	5499	11138	9403	
2	9064	3297	5757		
3	6329	2327	3992		
4	4976	1846	3121		
5	4154	1533	2612		
6	3666	1368	2288		
7	3295	1214	2070		
8	3032	1103	1918		
9	2791	969	1811		
10	2579	835	1734		
11	2531	832	1688		
12	2511	839	1661		
13	2342	684	1647		
14	2303	648	1643		
15	2289	635	1643		
16	2223	571	1641	1318	1160
				1030	920
				821	736
				654	579
				500	435
				379	315
				250	183
				110	31
17	2190	537	1642		
18	2170	518	1642		
19	2147	494	1642		
20	2123	470	1642		

por P_i para resolver o seu subdomínio de transporte, ou seja, montar as matrizes e executar o *solver*. Assim, a capacidade total de processamento dos processos P_i e P_{i+1} é dada por $C_i + C_{i+1}$ e o número total de células a serem processadas por ambos é dada por $N_i + N_{i+1}$. Tendo o número total de células de ambos os processadores envolvidos na transferência de carga e a capacidade total de processamento de ambas, o tempo para processamento do domínio total pode ser dado por:

$$T_f = \frac{N_i + N_{i+1}}{C_i + C_{i+1}}$$

sendo que nesse tempo cada um dos dois processadores envolvidos processará um número de células proporcional à sua disponibilidade de processamento. Para isso, o número de células a ser enviada pelo processador P_i é dado por:

$$N_{enviar} = (T_i - T_f) * C_i$$

e o número de células a ser recebida pelo processador P_{i+1} é dado por:

$$N_{receber} = (T_f - T_{i+1}) * C_{i+1}$$

O algoritmo descrito foi implementado e testado em diversas situações. Para a situação inicial, quando não é possível calcular a capacidade de processamento dos processadores porque apenas o processo P_0 possui células da malha refinada foi utilizado o mecanismo descrito na seção 7.9.

O primeiro teste foi realizado com 4 processadores, um por nodo, nos nodos L1 a L4. Foram efetuados 2000 ciclos utilizando um refinamento igual a 2 e o algoritmo RCB para o particionamento. Após a distribuição inicial da carga, foram executados alguns ciclos com cada processo rodando em um nodo diferente, de forma dedicada, sem outras aplicações compartilhando o mesmo nodo. A monitoração do tempo para montagem e solução das matrizes nos primeiros 1000 ciclos mostrou que o tempo utilizado era o mesmo em todos os ciclos, por todos os processos. Após os primeiros 1000 ciclos, foram disparadas outras 3 aplicações no nodo L4, onde era executado o processo P_3 da simulação, de forma que, havendo 4 processos sendo executados no nodo, o processo P_3 passou a dispor de aproximadamente metade da capacidade de processamento de um processador. Essa distribuição do uso da CPU foi confirmada pelo utilitário *top*. A análise do tempo de execução de cada ciclo do processo P_3 mostrou que, a partir do momento em que os outros processos passaram a compartilhar o nodo L4 o tempo de execução do subdomínio passou a sofrer variações grandes entre um ciclo e outro, chegando a variações de 100%. Uma possível razão para essa variação era a existência de operações de entrada e saída nos outros processos, durante as quais o processo P_3 teria maior disponibilidade de CPU. Para eliminar esse fator utilizou-se como aplicativo para rodar em paralelo a P_3, um processo sem nenhuma operação de entrada e saída. Entretanto, testes realizados dessa forma ainda apontaram variações significativas no tempo de execução entre um ciclo e outro.

Para reduzir a influência da variação do tempo de execução entre cada ciclo, passou-se a utilizar o tempo médio de execução dos 10 últimos ciclos como medida da carga, ao invés do tempo do ciclo anterior. Desta forma, pequenas variações de tempo entre um ciclo e outro são diluídas entre os diversos ciclos, reduzindo a possibilidade de que o processo de redistribuição de carga seja disparado por variações esporádicas no tempo de processamento.

Testes executados utilizando a média dos tempos de execução dos 10 últimos ciclos foram efetuados e mostraram um comportamento bem mais estável, havendo, entretanto, situações em que as variações de tempo eram tais que o mecanismo de redistribuição era disparado. Uma hipótese, não verificada, para explicar esse comportamento é que, como o tempo de execução de um subdomínio do transporte é muito pequeno, a ocorrência ou não de interrupção pelo escalonador do Sistema Operacional durante o processamento do subdomínio pode afetar o tempo de execução do mesmo.Testes utilizando um refinamento igual a 3, para os quais o tempo de processar um subdomínio era maior, mostraram um comportamento mais estável, atingindo uma distribuição estável de carga. A figura 7.7 mostra o comportamento da carga ao longo de 3000 ciclos. Após a distribuição inicial de carga, os 4 processadores ficaram atendendo exclusivamente à simulação. No ciclo 1000 foram disparados outros 3 processos, que não utilizavam operações de entrada e saída, no nodo L4. No ciclo de número 2000, os três processos foram descontinuados voltando

os 4 processadores a atender exclusivamente a simulação. A figura 7.8 mostra os tempos de execução dos ciclos de transporte durante a simulação.

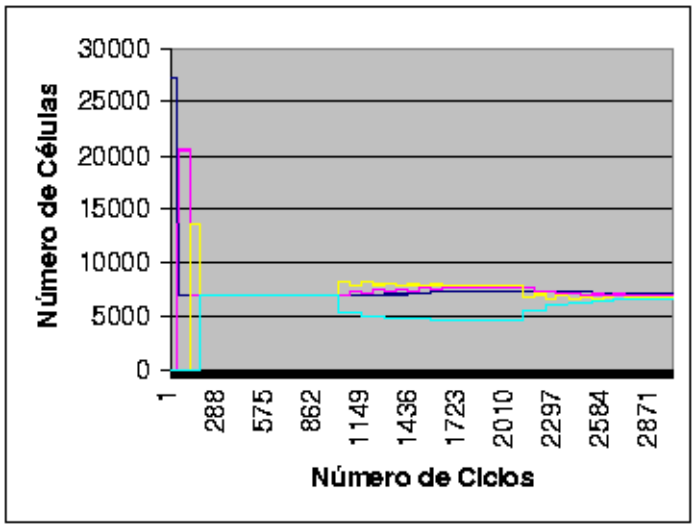


Figura 7.7: Comportamento da carga em nodo não dedicado

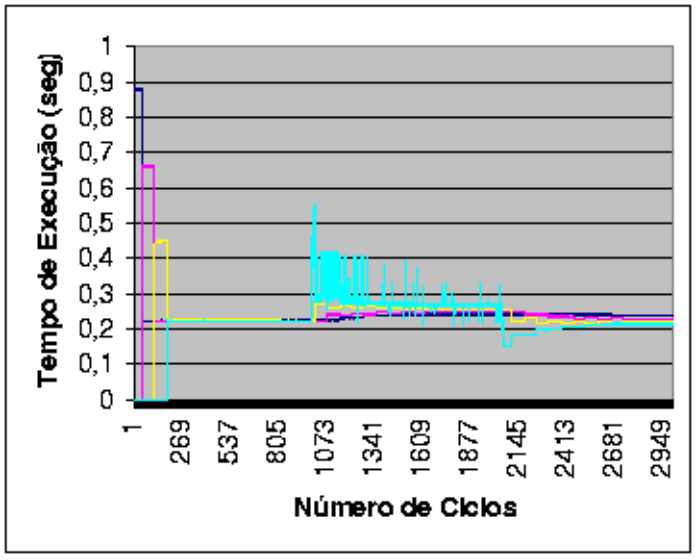


Figura 7.8: Comportamento do tempo de execução em nodo não dedicado

Pode-se observar que ao término da distribuição inicial da carga, aproximadamente no ciclo 200, todos os processadores têm a mesma carga (de aproximadamente 6500 células) e levam o mesmo tempo (aproximadamente 0.22 segundos) para processá-la, uma vez que os nodos são homogêneos e não há mais nenhum processo sendo executado. Aproximadamente no ciclo 1000 a execução de outros processos no nodo L4 faz com que a disponibilidade de processamento para a simulação caia pela metade, fazendo com que o tempo para processamento das 6500 células suba para 0.5 segundos. A partir daí ocorre a migração de células do processo no nodo L4 para os outros processos, estabilizando a carga novamente. Apesar da carga ter estabilizado, nota-se que o tempo de processamento no nodo L4 apresenta variações, decorrentes dos fatores discutidos anteriormente. No ciclo 2000 os processos que

compartilhavam o nodo L4 com a simulação foram descontinuados, ficando o nodo L4 novamente dedicado à simulação e fazendo com que o tempo de execução, que tinha se estabilizado em 0.27 caísse e fazendo com que carga migrasse novamente para L4. Após alguns ciclos a carga estabiliza numa situação próxima à situação inicial, havendo alguma diferença de carga entre os nodos devido à tolerância utilizada na difusão, o que não ocorria na distribuição inicial.

7.9 Solução exata e distribuição inicial

O mecanismo de difusão, conforme discutido na seção 3.2, é mais apropriado para situações onde os desbalanceamentos são pequenos. Na situação inicial do modelo, onde toda a malha refinada reside no processo 0, a difusão não se mostrou um mecanismo apropriado, sendo necessário um número muito grande de ciclos até que fosse atingida uma situação de boa distribuição da carga, a partir da qual a difusão se mostrasse um mecanismo eficiente.

Uma alternativa para o rebalanceamento quando há diferenças grandes de carga num particionamento por faixas como o utilizado no modelo é utilizar um mecanismo no qual cada processo possa calcular, a partir da informação da carga total de todos os outros processos, a quantidade exata de carga que deve ser enviada ou recebida dos vizinhos para chegar a uma distribuição ótima. Como foi visto na seção 3.2 a solução do problema de transferência de carga para chegar a um balanceamento ótimo em uma organização linear de processadores é única e a carga de cada processador pode ser calculada dividindo a carga total pelo número de processadores. Sabendo a quantidade de carga que deve estar à sua direita e à sua esquerda e conhecendo a distribuição da carga entre os processadores, cada processador tem condições de calcular a quantidade de carga que deve enviar ou receber para cada um dos dois processadores vizinhos. Assim, com uma operação de comunicação coletiva (*all-to-all*) cada processador passa a saber a carga de todos os outros e pode calcular quanta carga deve enviar e receber.

As restrições de largura mínima e número mínimo de células impostas aos subdomínios, entretanto, impedem que seja adotada essa solução, uma vez que seria necessário possuir informações sobre a geometria de todos os subdomínios para calcular a carga de cada um respeitando as restrições. O problema da distribuição inicial da carga, entretanto, é mais simples do que o problema geral da redistribuição da carga e pode ser resolvido de forma satisfatória sem o uso de comunicação global.

Na solução adotada, o processo P_0, que detém inicialmente toda a malha refinada, calcula o número de células N_0 que deverá manter, supondo que todos os processadores têm a mesma capacidade de processamento, dado por $N_0 = Total/Num\_Procs$. P_0 mantém, então, o número de células calculadas e envia as outras células para o processo P_1. O processo P_1, no ciclo seguinte, calcula o número de células que deve manter dado por $N_1 = N_1/(Num\_Procs - 1)$ e envia as células restantes, dadas por $N_1/(Num\_Procs - 1) * (Num\_Procs - 2)$ para o processo P_2 e assim por diante, até a carga estar toda distribuída. Essa situação inicial pode ser identificada pelos processos a partir da carga do vizinho à direita. Assim, a cada ciclo de redistribuição, cada processo P_n verifica se a carga do vizinho à direita é igual a zero e, se for, calcula quanto de sua carga deve ser transferida e a transfere. Esse esquema para a distribuição inicial da carga tem a vantagem de não necessitar de comunicação glo-

bal e de que pode ser executado juntamente com os ciclos normais de redistribuição, sem necessitar de um sincronismo global, uma vez que cada processo P_i, uma vez que tenha enviado sua parte da carga para o processo P_{i+1}, pode passar a executar os ciclos normais de rebalanceamento, sem ser necessário esperar o fim de todo o processo de distribuição inicial.

O mecanismo descrito foi implementado e foi testado apresentando um ganho de desempenho significativo em relação ao uso do mecanismo de difusão para a inicialização. A figura 7.9 mostra a distribuição inicial da carga para 4 processadores e a tabela 7.17 mostra os tempos obtidos para uma simulação de 10.000 ciclos com refinamento igual a 2 e particionamento pelo RCB.

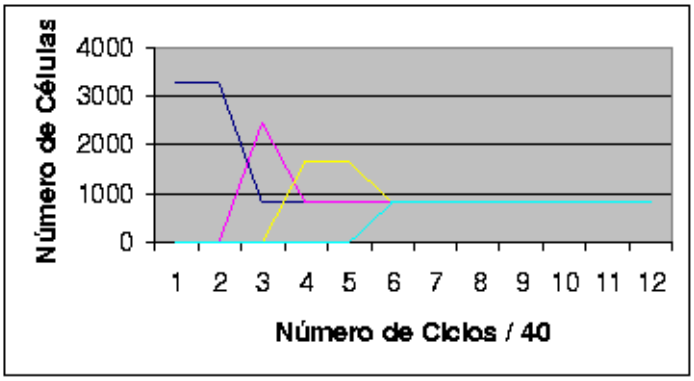


Figura 7.9: Distribuição inicial da carga para 4 processadores

7.10 Efeito do número de camadas e da batimetria

Os resultados até aqui apresentados foram baseados em chão plano com uma única camada, ou seja, a profundidade de cada célula é a mesma em todo o domínio. Nessa seção será feita uma rápida discussão de como o aumento do número de camadas e o uso da batimetria real afetam o desempenho do modelo. Os testes foram realizados apenas sobre o cálculo a hidrodinâmica, por ser o que maior influência sofre do aumento do número de camadas, uma vez que ocorre em todo o domínio.

Nos testes foram executados 1000 ciclos sobre o Guaiba 200 com refinamento igual a 2, sendo utilizado o RCB para o particionamento. As tabelas 7.18, 7.19 e 7.20 mostram o tempo de execução do modelo respectivamente em um único processador, em dois processadores e em oito processadores, com o número de camadas variando de 1 a 8, com chão plano e com batimetria real. As colunas 2 e 3 mostram o tempo total e o tempo utilizado na montagem das matrizes da hidrodinâmica na execução com chão plano e as colunas 4 e 5 mostram os mesmos dados referentes à execução com batimetria real. Nas colunas 3 e 5, que mostram o tempo utilizado na hidrodinâmica, é considerado o tempo do cálculo das matrizes de coeficientes, resolução do sistema de equações e atualização das variáveis de velocidade e nível em todo o domínio. Como nesse teste foi executada unicamente a hidrodinâmica, o tempo restante da simulação é utilizado na comunicação entre processos vizinhos e no cálculo de algumas variáveis necessárias para a montagem das matrizes dos sistemas em todo o domínio (3D). Nas execuções com 2 a 8 processadores, como o tempo da hidrodinâmica é diferente para cada um dos processadores envolvidos, a tabela apresenta o maior e o menor dos tempos para a execução da hidrodinâmica.

Tabela 7.17: 10000 ciclos com malhas acopladas com REF=2, particionamento utilizando RCB e distribuição exata da carga

Número de processos	Tempo total	Tempo total da hidrodinâmica	Tempo do Solver da Hidrodinâmica		Tempo total do transporte	Tempo do solver do transporte	
1	8657	2680	1842		5971	5011	
2	4618	1591	1072	1129	3022	2498	2414
3	3210	1118	747	796	2088		
4	2524	894	572	638	1625		
5	2113	754	512	544	1354		
6	1858	678	424	495	1175		
7	1675	614	395	451	1055		
8	1493	520	323	377	967		
9	1391	482	294	347	904		
10	1318	452	264	323	858		
11	1250	419	255	306	823		
12	1179	374	223	263	798		
13	1133	348	205	247	780		
14	1175	404	194	306	763		
15	1126	356	163	265	762		
16	1100	332	152	243	763	440	434
						412	391
						372	352
						335	320
						305	294
						283	274
						265	249
						11	0
17	1156	388	168	292	762		
18	1028	262	140	183	757		
19	1017	253	121	176	757		
20	1005	243	121	169	755	428	427
						406	386
						368	351
						334	319
						304	293
						283	274
						265	259
						35	0
						0	0
						0	0

O objetivo de apresentar esses dados é mostrar o desbalanceamento da malha da hidrodinâmica em cada situação.

O uso de batimetria real, do ponto de vista de desempenho e balanceamento de carga, faz com que a quantidade de dados transferidos entre subdomínios vizinhos varie com a profundidade da região da fronteira entre os subdomínios, acrescentando

Tabela 7.18: Efeito do número de camadas com 1 processador

Número de camadas	Tempo total com chão plano	Tempo do *solver* da hidrodinâmica com chão plano	Tempo total com batimetria	Tempo do *solver* da hidrodinâmica com batimetria
1	269	185	277	193
2	358	215	307	198
3	436	236	338	204
4	535	274	390	216
5	641	310	416	219
6	722	339	435	220
7	820	376	465	226
8	929	411	493	227

Tabela 7.19: Efeito do número de camadas com 2 processadores

Número de camadas	Tempo total com chão plano	Tempo do *solver* da hidrodinâmica com chão plano		Tempo total com batimetria	Tampo do *solver* da hidrodinâmica com batimetria	
1	151	106	106	159	114	114
2	193	117	117	171	112	113
3	233	128	128	187	116	116
4	285	148	148	215	121	123
5	345	171	172	228	122	124
6	384	186	186	239	124	125
7	435	206	207	252	128	128
8	498	226	228	268	129	129

Tabela 7.20: Efeito do número de camadas com 8 processadores

Número de Camadas	Tempo Total com chão plano	Tempo do solver da Hidrodinâmica com chão plano		Tempo total com batimetria	Tampo do solver da Hidrodinâmica com batimetria	
1	44	27	29	46	28	31
2	58	30	34	50	28	31
3	71	32	37	56	29	32
4	86	37	43	66	30	36
5	105	44	52	69	30	35
6	119	48	58	72	31	35
7	134	53	64	78	32	37
8	152	57	71	81	32	37

uma terceira dimensão a ser levada em conta no cálculo do corte de arestas entre subdomínios. Além disso, células com uma profundidade maior e maior número

de camadas têm um maior custo computacional para o seu cálculo, tanto na fase da hidrodinâmica, por ocasião do cálculo da velocidade vertical em cada coluna, quanto na fase do transporte, que é resolvido em cada coluna vertical sendo gerado um sistema com um número de equações igual ao número de camadas da coluna. Cabe ressaltar que o sistema gerado é tridiagonal e resolvível em tempo linear pelo algoritmo de Thomas.

8 CONCLUSÕES E TRABALHOS FUTUROS

O particionamento do domínio e balanceamento de carga são pontos fundamentais para o bom desempenho em modelos computacionais paralelos de sistemas físicos como o descrito neste trabalho. Inúmeros trabalhos vêm sendo publicados nesta área, para diferentes tipos de modelos e arquiteturas. Neste trabalho foram apresentados os mecanismos de particionamento e balanceamento dinâmico de carga desenvolvidos para um modelo paralelo de circulação e transporte de substâncias do Lago Guaíba desenvolvido no Instituto de Informática da UFRGS. Foram apresentados rapidamente o modelo numérico e os métodos utilizados em sua solução e paralelização, apresentados em detalhes em (RIZZI, 2002). Este modelo agrega diversas características como refinamento local e o uso de malhas adaptativas que tornam necessário o uso de mecanismos eficientes de particionamento de domínio e de balanceamento dinâmico de carga. O desenvolvimento desses mecanismos foi o objetivo desta tese.

8.1 Trabalhos já publicados

As diferentes versões do modelo desenvolvido neste trabalho bem como diversos trabalhos relacionados foram apresentadas em vários eventos internacionais, nacionais e regionais de processamento paralelo. Alguns desses eventos são apresentados a seguir:

- No VECPAR 2000, em Porto, Portugal , no artigo intitulado *"Fluvial Flow of the Guaiba River - A parallel solution for the Shallow Water equations model"* (RIZZI et al., 2000), foi descrito e apresentado o modelo de hidrodinâmica discretizado utilizando a abordagem de Casalas (BORCHE CASALAS, 1985) e resolvido utilizando o método de Thomas.

- No *Symposium on Computer Architecture and High Performance* (SBAC-PAD), em Águas de São Pedro, São Paulo, no artigo intitulado *"PC Cluster Implementation of a Mass Transport two-dimensional Model"* (DORNELES et al., 2000), foi descrito o modelo de hidrodinâmica discretizado utilizando a abordagem de Casulli com decomposição de domínio e resolvido em cada subdomínio utilizando o algoritmo do Gradiente Conjugado, bem como o modelo de transporte de substâncias resolvido utilizando o método de Thomas.

- No Workshop de Sistemas Distribuidos y Paralelismo, em Santiago, no artigo intitulado *"Parallel Solution for Shallow Water Equation using the Data*

Decomposition" (DORNELES et al., 2000), foi descrito e apresentado um modelo da hidrodinâmica discretizado via diferenças centradas em uma grade não coincidente (*staggered*) e resolvido via Gradiente Conjugado paralelo sobre todo o domínio. O artigo descreveu em detalhes a paralelização do algoritmo do Gradiente Conjugado, bem como os resultados obtidos com a versão paralela.

- No *Symposium on Computer Architecture and High Performance*(SBAC-PAD), em Pirenópolis, Goiás, no artigo intitulado *"A h-Refined Krylov-Schwarz Solution for Hydrodynamics and Mass Transport in a PC Cluster"*,(DORNELES et al., 2001), foi apresentado o primeiro modelo de hidrodinâmica e transporte usando uma malha refinada.

- No VECPAR 2002, em Porto, Portugal, no artigo intitulado *"Parallel solution in PC clusters by the Schwarz domain decomposition for three-dimensional hydrodynamics models"*, (RIZZI et al., 2002), foi descrito e apresentado o modelo de hidrodinâmica 3D.

Além dos trabalhos citados, também foram publicados os seguintes trabalhos, em autoria ou co-autoria: (RIZZI; DORNELES; DIVERIO, 1999), (GALANTE et al., 2002), (CARVALHO et al., 2002), (ZEFERINO et al., 2000), (PICININ et al., 2001), (PICININ et al., 2002), (PICININ et al., 2002). Diversos dos trabalhos citados apresentam as soluções utilizadas para particionamento estático do domínio da hidrodinâmica e do transporte. Os mecanismos de balanceamento dinâmico aqui propostos, implementados e avaliados foram apresentados pela primeira vez na proposta de tese (DORNELES, 2001) e um artigo específico sobre uma das abordagens implementadas (*Dynamic Load Balancing in PC Clusters: An Application to Multi-Physics Problems*) foi submetido ao SBAC-PAD de 2002. Entretanto, apesar de os avaliadores terem reconhecido o mérito científico e a relevância do trabalho, o mesmo foi rejeitado por questões de forma tendo sido sugerido que o artigo fosse dividido em outros dois, um deles abordando problemas multi-fase e o outro abordando balanceamento dinâmico de carga. Estão sendo previstas para 2003 submissões de artigos relatando os resultados obtidos para a *Parallel Computing* (Elsevier Science) e *Applied Numerical Mathematics*, além de submissões para eventos na área de processamento paralelo (VECPAR, SBAC-PAD e WSDP).

8.2 Trabalhos já desenvolvidos ou sendo desenvolvidos pelo grupo

A partir do trabalho descrito aqui e em (RIZZI, 2002), diversos outros trabalhos foram ou estão sendo desenvolvidos onde tópicos relacionados são pesquisados com mais profundidade. Os trabalhos já concluídos até o momento são os seguintes:

- Ana Paula Canal, mestranda da turma de 1998, desenvolveu, sob orientação do Prof. Dr. Tiarajú Asmuz Diverio, a dissertação de mestrado intitulada "Paralelização de métodos de resolução de sistemas lineares esparsos com o Deck em um cluster de PCs", onde implementou versões paralelas para o Método do Gradiente Conjugado e o Método de Thomas utilizando a biblioteca DECK;

- Marcos André Künzel Palha, aluno de graduação da Universidade de Caxias do Sul, desenvolveu como trabalho de conclusão (PALHA, 2000)um estudo sobre ferramentas para *tracing* de programas paralelos, utilizando como estudo de caso o Hidra 2D;

- Elias Araujo, mestrando da turma remota de Rolândia, desenvolveu, sob orientação do Prof. Dr. Tiarajú Asmuz Diverio, a dissertação de mestrado (CARVALHO, 2002) no desenvolvimento de algoritmos de mapeamento e modelagem do *cluster* como um grafo para aplicação dos algoritmos de mapeamento;

- Delcino Picinin Junior, mestrando da turma de 2001, desenvolveu, sob orientação do Prof. Dr. Tiarajú Asmuz Diverio, um trabalho sobre a paralelização do método do Gradiente Conjugado utilizando *threads* (PICININ, 2001) e em sua dissertação (PICININ, 2003) de mestrado, também orientado pelo Prof. Tiarajú, implementou e analisou a paralelização dos algoritmos do Gradiente Conjugado e GMRES utilizando MPI, DECK e Pthreads;

- Leandro Luis Puerari dos Santos, aluno de graduação da Universidade de Caxias do Sul, desenvolveu como trabalho de conclusão (SANTOS, 2001) um levantamento sobre algoritmos de particionamento de grafos, análogo ao presente neste trabalho, incluindo a implementação dos algoritmos LND e Fidducia-Mattheises;

- André Luis Martinotto, mestrando do PGCC da turma de mestrado de 2003 e ex-aluno de graduação da UCS, onde desenvolveu o Trabalho de Conclusão de Curso intitulado "Paralelização de Métodos Numéricos de Resolução de Sistemas Esparsos de Equações Utilizando MPI e Pthreads"(MARTINOTTO, 2001) e atualmente está trabalhando, sob orientação do Prof. Tiarajú, na implementação do método Krylov-Schwarz sobre o HIDRA-3D.

8.3 Conclusões

Neste trabalho foi estudado o problema de particionamento de domínio em problemas multi-fase em *clusters* de máquinas multiprocessadas, bem como mecanismos de difusão para manutenção de balanceamento de carga. Os mecanismos foram implementados sobre modelos 3D de hidrodinâmica e transporte de poluentes, desenvolvido em conjunto com o doutorando Rogério Rizzi, orientando do Prof. Tiarajú.

Aplicações multi-fase, como a abordada nesse trabalho, apresentam características que tornam bem mais complexo o seu particionamento e mapeamento. Neste trabalho foram apresentadas duas abordagens para o particionamento e balanceamento dinâmico em problemas multi-fase. Ambas abordagens foram implementadas e seu desempenho foi avaliado. Como foi mostrado na seção 3.1, a implementação de um esquema de balanceamento de carga em uma aplicação onde a distribuição da carga está continuamente sendo alterada, envolve os seguintes passos:

- Implementação de mecanismos de avaliação da carga, podendo ser esta analítica, por medição ou híbrida;

- Implementação de um mecanismo, centralizado ou distribuído, de cálculo da relação custo-benefício de se proceder a uma redistribuição e de decisão;

- Cálculo da carga a ser transferida;
- Transferência de carga;

Na fase da elaboração da proposta de tese foram estudadas duas abordagens de redistribuição de carga, que foram a de *scratch-remap* e a de difusão. Considerando que na aplicação abordada neste trabalho a malha da hidrodinâmica permanece inalterada e a malha do transporte de substâncias não sofre alterações bruscas, considera-se a difusão um mecanismo mais apropriado para a manutenção do balanceamento da carga. Entretanto, para a distribuição inicial da malha refinada a partir do Processo 0, como ocorrem transferências de carga significativas até que o domínio esteja distribuído de forma balanceada, a solução direta para o problema da distribuição inicial da carga, implementada e descrita na seção 7.9, apresentou melhores resultados, pelo menos até o número de processos utilizados (20 processos, na maior parte dos testes).

Parte desse trabalho consistiu no estudo de soluções para balanceamento dinâmico de carga em problemas multi-fase, mais especificamente no balanceamento dinâmico em um modelo de circulação e transporte de substâncias sobre duas malhas acopladas, e na implementação de duas soluções possíveis (além de uma terceira implementação sem mecanismos de balanceamento dinâmico para efeito de comparação). Os resultados obtidos com o modelo sem balanceamento comprovaram a necessidade de mecanismos de balanceamento dinâmico nesta classe de modelos.

Duas soluções de balanceamento dinâmico foram implementadas no modelo. Na primeira, as duas malhas são particionadas independentemente e os dados necessários para a malha refinada são buscados, a cada ciclo, do processo que os detém; na segunda, inicialmente a malha refinada e a parte da malha da hidrodinâmica associada a ela são particionadas com um particionamento por faixas e na segunda etapa é particionada a parte da malha da hidrodinâmica onde não ocorre transporte.

Conforme mostrado na subseção 7.4, ambas abordagens apresentaram ganhos de desempenho satisfatórios, sendo que em alguns casos a abordagem do particionamento independente das malhas apresentou melhor resultado e em alguns casos o particionamento acoplado das malhas apresentou melhor resultado.

O particionamento por faixas da malha refinada resolveu satisfatoriamente o problema de se identificar, a partir das coordenadas de cada célula da malha refinada, a qual processo a mesma pertencia, permitindo que durante a expansão da malha cada processador soubesse a qual processador pertencia cada célula recém alocada e evitando que dois processadores ficassem responsáveis pelo processamento da mesma célula. Sabe-se que o corte de arestas no particionamento em faixas é excessivo, mas não se encontrou uma solução para a expansão da malha refinada utilizando um particionamento irregular. De qualquer forma, as medidas realizadas mostram que o ganho de desempenho no cálculo do transporte bem como no desempenho geral do modelo foi bem satisfatório.

O desempenho de um esquema de balanceamento dinâmico como o descrito neste trabalho é bastante dependente de alguns parâmetros como a freqüência com que a distribuição de carga é analisada pelos processos, o percentual da diferença de carga a ser transferido e a diferença de carga a partir da qual é disparado o processo de redistribuição. No particionamento por faixas como o utilizado para a malha refinada, mesmo tolerâncias pequenas na diferença entre as cargas de processos vizinhos podem se acumular de forma que a diferença de carga entre o processo P_0

e o último processo seja significativa.

Também foram feitos testes para avaliar o efeito do percentual de diferença de carga a ser transferido na redistribuição da carga. Inicialmente os testes foram realizados utilizando um percentual de 40%, mas testes com diversos valores do percentual mostraram que percentuais mais altos convergiam num número de ciclos significativamente menor, ocorrendo oscilações na distribuição da carga mas convergindo mais rapidamente (considerando o tempo de execução) a uma distribuição balanceada. Isso ocorreu até um percentual de aproximadamente 70%, a partir do qual o esquema não mais convergiu.

Foi utilizada uma freqüência fixa de balanceamento, na qual as trocas de mensagens necessárias para a redistribuição da carga são feitas a cada 30 ciclos. Não foram feitos testes de como essa freqüência afeta o desempenho do modelo ou se um esquema com freqüência variável traria melhor desempenho ao modelo. Entretanto, como essas trocas envolvem apenas comunicação com processadores que detenham subdomínios vizinhos, seu custo é baixo, quando comparado com a quantidade de mensagens trocadas em todos os ciclos para atualização dos dados das fronteiras. Além disso, um mecanismo de freqüência variável necessitaria de comunicação global uma vez que o esquema de redistribuição implementado exige que todos os processadores executem o mecanismo de redistribuição no mesmo ciclo, e é possível que, com um número grande de processadores, o custo da comunicação global não compensaria o ganho obtido com a redução do número de vezes em que o mecanismo de redistribuição seria disparado, mas isso ainda está para ser avaliado.

Os testes realizados em máquinas duais mostraram que quando dois processos são executados em um nodo dual, há uma perda significativa de desempenho (para os testes realizados essa perda ficou em 10%) em relação à execução de um único processo no mesmo nodo, provavelmente devido à disputa pelo acesso à memória. Essa perda é tal que faz com que a execução de dois processos da mesma aplicação em nodos diferentes apresente um desempenho melhor que a execução dos mesmos processos no mesmo nodo, mesmo utilizando comunicação entre os processos através da memória compartilhada. Esses resultados mostram também que, para aplicações onde ocorre uso intensivo de memória como modelos de simulação de fenômenos físicos, pode ser mais vantajoso utilizar *clusters* de máquinas monoprocessados do que *clusters* de máquinas duais. Testes feitos com aplicações com menor uso de memória poderiam mostrar se ocorre a mesma perda de desempenho nessa situação.

Apesar de o tema desse trabalho ser os algoritmos de particionamento e balanceamento dinâmico, para que os mesmos pudessem ser aplicados foi necessário o desenvolvimento do modelo de hidrodinâmica e transporte utilizado ao longo desse trabalho em um trabalho conjunto com o doutorando Rogerio Rizzi. O modelo, utilizado em ambos os trabalhos e descrito em maiores detalhes em (RIZZI, 2002), utiliza estruturas de dados e procedimentos razoavelmente complexos para gerenciamento das fronteiras entre os subdomínios. Para seu desenvolvimento foram necessários alguns milhares de horas de programação e depuração, além da leitura de numerosas referências sobre métodos numéricos e paralelização dos mesmos, formas de armazenamento de matrizes irregulares, decomposição de domínio, particionamento de grafos e balanceamento de carga. Parte do resultado desse estudo se encontra nos artigos listados na seção 8.1, e nesse trabalho, na forma do modelo desenvolvido.

Um dos aspectos mais críticos neste trabalho foi o tratamento correto das inúmeras situações de dependências entre células dos subdomínios envolvidos nas transferências.

Para simplificar esse tratamento e reduzir o número de mensagens envolvidas foram adotadas algumas medidas simplificadoras, como não efetuar a expansão da malha do transporte em processos vizinhos no mesmo ciclo, mas que não afetam de forma significativa o desempenho do modelo ou a qualidade dos resultados.

Uma preocupação presente em todos os momentos do desenvolvimento desse modelo é a eficiência computacional ao executar os modelos. A maior parte das operações sobre o domínio (incluindo os *solvers* utilizados) tem complexidade linear, conforme mostrado na seção 4.1.6 o que assegura um bom desempenho ao código obtido.

Apesar da preocupação com a complexidade (no sentido de custo computacional) do código desenvolvido, há espaço para um grande número de otimizações no código de forma a melhorar seu desempenho. Ao terminar a primeira versão do modelo 2D, foram dedicadas algumas horas ao perfilamento do código seqüencial e constatou-se que o cálculo das matrizes de coeficientes dos sistemas eram responsáveis por uma parcela significativa do tempo de processamento. Para avaliar o quanto pode ser obtido de melhora de desempenho neste tipo de código, foram efetuadas algumas otimizações locais como identificação de sub-expressões reutilizadas e armazenamento das mesmas. As otimizações resultantes reduziram o tempo de execução do código seqüencial em mais de 50% resultando, entretanto, em perda de legibilidade e manutenabilidade. A versão atual do código 3D com os algoritmos de balanceamento dinâmico não possui as otimizações, pois para o desenvolvimento de um modelo como esse, a legibilidade do código é essencial, além de não ser objetivo do trabalho obter o modelo seqüencial mais rápido possível. Entretanto, os experimentos efetuados sobre o código 2D mostram que podem ser obtidas melhoras significativas no desempenho das versões posteriores dos modelos 2D e 3D.

Durante o desenvolvimento deste trabalho e da tese de Rogerio Rizzi, foram encontradas referências a 35 modelos de circulação e transporte de substâncias (RIZZI, 2002), sendo que a grande maioria deles se resume a circulação, sem ter o modelo de transporte de substâncias acoplado. Nenhum dos modelos encontrados incorpora mecanismos de balanceamento dinâmico de carga como os apresentados nesse trabalho. Um levantamento desses modelos, com uma tabela comparativa das características dos 35 modelos encontrados está para ser publicado como um relatório de pesquisa (RIZZI; DORNELES; DIVERIO, 2003).

Na maior parte do trabalho de implementação dos mecanismos de balanceamento dinâmico, foi usado como métrica para avaliação de carga o número de células em cada processador. Essa medida da carga, por não depender de fatores externos à aplicação, mostrou-se uma boa escolha para as etapas de implementação e depuração do modelo. Entretanto, o balanceamento resultante da escolha dessa medida não foi totalmente satisfatório, o que pode ser observado na comparação do tempo de montagem e solução das matrizes, tanto na fase de hidrodinâmica quanto na fase de transporte. Algumas razões para isso foram discutidas, como a geometria do domínio, que podia causar grandes variações no número de células de sobreposição e conseqüentemente no tempo de processamento dos subdomínios. O uso do tempo de processamento dos subdomínios como medida da carga dos processadores, inicialmente feita apenas para avaliar o uso dos mecanismos em ambientes compartilhados, mostrou-se também uma escolha mais apropriada como medida de avaliação de carga, mesmo em *clusters* homogêneos de uso dedicado.

8.3.1 Contribuições desse trabalho

A contribuição maior desse trabalho é a integração de diversos algoritmos e soluções de particionamento e balanceamento dinâmico em um modelo real, que com algumas poucas alterações e devidamente calibrado pode ser utilizado fornecendo resultados realísticos com um bom desempenho. Algumas contribuições pontuais desse trabalho são:

- Desenvolvimento do modelo 3-D (com Rizzi)
- Proposta de duas abordagens para balanceamento dinâmico em problemas multi-fase, bem como a implementação e análise de ambas abordagens;
- Análise do desempenho dos diversos particionadores no modelo implementado para uma camada e para diversas camadas, considerando batimetria real
- Adaptação do algoritmo de refinamento local para a seleção das células
- Avaliação do mecanismo de balanceamento em situações de compartilhamento de carga bem como em máquinas biprocessadas
- Análise do efeito dos diversos parâmetros do mecanismo de balanceamento no desempenho do mesmo
- Formação de Recursos Humanos : Como foi mostrado na seção 8.2, o desenvolvimento deste trabalho serviu de motivação para o desenvolvimento de diversos trabalhos de conclusão de curso e dissertações de mestrado, bem como contou com a participação de numerosos bolsistas, contribuindo assim, de forma significativa, para a formação de recursos humanos na área de processamento paralelo e de alto desempenho.

8.4 Trabalhos Futuros

Para que o modelo desenvolvido neste trabalho e em (RIZZI, 2002) se torne operacional ainda há diversos aperfeiçoamentos que devem ser incorporados. Rizzi (RIZZI, 2002) cita, entre outros:

- Desenvolvimento de um modelo de parametrização de turbulência
- Implementação de um esquema que suporte alagamento e secamento de células
- Utilização de outro sistema de coordenadas verticais
- Discretização das EDPs utilizando volumes finitos

Uma vez obtido um modelo operacional, há diversos pontos que foram levantados nesse trabalho que podem ser trabalhados para melhorar o desempenho do modelo, além da otimização do código através do uso de perfiladores, como foi sugerido na seção anterior. Alguns desses pontos a serem trabalhados são:

- Implementação de outros esquemas de difusão, como esquemas de segunda ordem ou o esquema de difusão de Hu e Blake, que apresenta uma melhor convergência.

- No capítulo 6 foi justificado o uso do particionamento por faixas para a malha refinada. Mas estudos adicionais visando o uso de particionamento irregular para transporte poderiam ser compensatórios, considerando o percentual do esforço computacional total que é dispendido no cálculo do transporte.

- Outro ponto a ser explorado é o particionamento do modelo levando-se em conta o número de camadas em cada posição do domínio. Os particionamentos utilizados no modelo, até o momento, modelam a vizinhança entre duas posições contíguas no domínio através de uma aresta de peso unitário. Entretanto, a quantidade de dados trocada através da fronteira entre dois processos depende do número de camadas das posições envolvidas e um modelo mais preciso dessa vizinhança é uma aresta com peso proporcional ao número de camadas.

- Outro ponto a ser melhorado é a seleção das células a serem migradas no particionamento irregular da hidrodinâmica na abordagem das malhas acopladas. Como foi colocado na seção 6.3.3, a seleção de cada célula a ser migrada baseia-se no custo da célula, sem levar em conta as transferências subseqüentes, o que, se fosse feito, aumentaria muito o custo computacional da seleção das células. Entretanto, é possível que um algoritmo que leve em conta apenas as n próximas células a serem migradas para o cálculo do ganho consiga uma qualidade de corte melhor a um custo ainda compensatório.

- Como foi descrito na seção 6.3.1, quando a concentração de poluente em algumas regiões do domínio desce abaixo de um valor limite e deixa de ter efeito significativo na simulação, é conveniente que essa parte da malha seja desalocada, uma vez que sua manutenção no domínio gera custo computacional. Por restrições de tempo, essa desalocação foi implementada apenas na primeira abordagem, apresentando bons resultados, mas sua implementação nas outras duas abordagens não deve apresentar maiores dificuldades;

- Como foi dito na seção anterior, algumas restrições em relação à transferência da carga foram colocadas no modelo de forma a reduzir o número de processos envolvidos. Um estudo sobre formalismos para especificação das seqüências de mensagens necessárias nas diferentes situações poderia levar a um esquema de transferência mais eficiente, que envolvesse menos mensagens, e no qual não fosse necessário impor restrições

- Foram efetuados diversos testes sobre o efeito dos parâmetros da difusão na convergência dos algoritmos e no desempenho do modelo. Entretanto, ainda há pontos que podem ser melhorados, como a freqüência de disparo do mecanismo de rebalanceamento e a implementação de um mecanismo de freqüência variável de disparo do mecanismo de balanceamento;

- Por razões de disponibilidade de recursos computacionais, os testes com os mecanismos de balanceamento foram limitados a 20 processadores, pertencentes a 10 nodos duais. A avaliação do desempenho sobre esses processadores mostrou que, para o tamanho de domínio utilizado nos testes, não seria obtido um ganho maior de desempenho aumentando o número de processadores. Seria interessante, entretanto, efetuar medidas com maior número de processadores para domínios de maior refinamento para avaliar a escalabilidade. Uma

possível métrica para avaliar a escalabilidade com maior número de processadores seria o uso do *scaled speedup* (BAKER; SMITH, 1996), no qual se mantem fixa a relação entre o tamanho do problema e o número de processadores, métrica aplicável em problemas como o modelo desenvolvido neste trabalho, em que o tamanho do domínio pode crescer tanto pelo aumento da área geográfica coberta pelo modelo quanto pelo refinamento da malha usada, com resultados mais precisos.

REFERÊNCIAS

ALURU, S.; SEVILGEN, F. Parallel Domain Decomposition and Load Balancing Using Space-filling Curves. In: HIGH PERFORMANCE COMPUTING, 1997, Bangalore, India. **Proceedings...** [S.l.: s.n.], 1997. Disponível em: <http://www.serc.iisc.ernet.in/hipc/hipc97>. Acesso em: jan. 2001.

ARAVINTHAN, V.; JOHNSON, S. P.; MCMANUS, K.; WALSHAW, C.; CROSS, M. Dynamic Load Balancing for Multi-Physical Modelling using unstructured meshes. In: INTERNATIONAL CONFERENCE ON DOMAIN DECOMPOSITION METHODS, 11., 1999. **Proceedings...** [S.l.: s.n.], 1999.

BAKER, L.; SMITH, B. J. **Parallel Programming**. New York: McGraw-Hill, 1996. 381p.

BARRETO, M. **DECK** : um ambiente para programação paralela em agregados de multiprocessadores. 2000. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

BARRETO, M. E.; ÁVILA, R. B.; OLIVEIRA, F.; CASSALI, R.; NAVAUX, P. DECK: an environment for parallel programming on clusters of multiprocessors. In: SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING, SBAC-PAD, 12., 2000, São Pedro, SP. **Proceedings...** São Carlos: UFSCAR, 2000.

BHARGAVA, R.; FOX, G.; OU, C.; RANKA, S.; SINGH, V. **Scalable Libraries for Graph Partitioning**. Disponível em: <http://www.npac.syr.edu/NPAC1/PUB/ranka/part/part.html>. Acesso em: out. 2000.

BISWAS, R.; OLIKER, L.; SOHN, A. Global Load Balancing with Parallel Mesh Adaption on Distributed-memory Systems. In: SUPERCOMPUTING, 1996, Pittsburg, USA. **Proceedings...** [S.l.: s.n.], 1996.

BODEN, N.; AL. et. Myrinet: a gigabit-per-second local-area-network. **IEEE Micro**, Los Alamitos, v.15, n.1, p.29–36, Feb. 1995.

BOILLAT, J. E. Load Balancing and Poisson Equation in a Graph. **Concurrency - Practice and Experience**, [S.l.], v.2, n.4, p.289–313, 1990.

BORCHE CASALAS, A. **Modelo Matemático da Correntologia do Estuário do Rio Guaíba**. Porto Alegre: IPH/UFRGS, 1985. (Recursos hídricos. Publicação, n.12).

CAMPOS, L. M.; SCHERSON, I. Rate of Change Load Balancing in Distributed and Parallel Systems. **Parallel Computing**, [S.l.], v.26, n.9, p.1213–1230, 2000.

CANAL, A. P. **Paralelização de Métodos de Resolução de Sistemas Lineares Esparsos com o DECK em um cluster de PCs**. 2001. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

CARISSIMI, A. S. **ATHAPASCAN-0**: exploitation de la multiprogrammation légère sur grappes de multiprocesseurs. 1999. Tese (Doutorado em Ciência da Computação) — Institut National Polytechnique de Grenoble, Grenoble, França.

CARVALHO, E. C. A. **Particionamento de Grafos de Aplicações e Mapeamento em Grafos de Arquiteturas Heterogêneas**. 2002. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

CARVALHO, E. C. A.; DORNELES, R. V.; RIZZI, R. L.; DIVERIO, T. A. Modelagem de um Cluster Heterogêneo para ser utilizado em Particionamento e Mapeamento de Domínio de Aplicações Paralelizáveis Baseado em Programas Específicos de Benchmarks. In: WPERFORMANCE, 1., 2002, Florianópolis, SC. **Anais...** Florianópolis:SBC, 2002. p.12.

CASAVANT, T. L.; KUHL, J. G. A Taxonomy of Scheduling in General-Purpose Distributed Computing System. **IEEE Transactions on Software Engineering**, [S.l.], v.14, n.2, p.141–154, 1988.

CASULLI, V. Semi-Implicit Finite Difference Methods for the Two-Dimensional Shallow Water Equations. **Journal of Computational Physics**, [S.l.], v.86, p.56–74, 1990.

CATABRIGA, L. **Estudo de Pré-Condicionadores para o Método GMRES usando Estruturas de Dados por Arestas**. Rio de Janeiro, RJ: UFRJ, 1998. Seminário de Qualificação ao Doutorado.

CERMELE, M.; COLAJANNI, M.; TUCCI, S. Check-Load Interval Analysis for Balancing Distributed (SPMD)Applications. In: INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED PROCESSING TECHNIQUES AND APPLICATIONS, PDPTA, 1997, Las Vegas, NV. **Proceedings...** [S.l.: s.n.], 1997. p.432–441.

CHAN, T.; SHAO, J. P. **Parallel Complexity of Domain Decomposition Methods and Optimal Coarse Grid Size**. Disponível em: <ftp://ftp.math.ucla.edu/pub/camreport/cam94-15.ps.gz>. Acesso em: maio 2001.

CHEN, J.; TAYLOR, C. Mesh Partitioning for Distributed Systems: exploring optimal number of partitions with local and remote communications. In: SIAM CONFERENCE ON PARALLEL PROCESSING FOR SCIENTIFIC COMPUTING, 9., 1999, San Antonio, Texas. **Proceedings...** [S.l.: s.n.], 1999.

CYBENKO, G. Load Balancing for Distributed Memory Multiprocessors. **Journal of Parallel and Distributed Computing**, [S.l.], v.7, p.279–301, 1989.

DEMMEL, J. **CS267 - Applications of Parallel Computers. Lecture 23**: load balancing and scheduling. Disponível em: <http://www.cs.berkeley.edu/~demmel/cs267_Spr99>. Acesso em: jun.2001.

DEMMEL, J. **CS267 - Applications of Parallel Computers - Spring 1996**. Disponível em: <http://www.cs.berkeley.edu/~demmel/cs267>. Acesso em: dez. 2000.

DIEKMANN, R.; MONIEN, B.; PREIS, R. **Load Balancing Strategies for Distributed Memory Machines**. Disponível em: <http://www.uni-paderborn.de/fachbereich/AG/monien/PUBLICATIONS/POSTSCRIPTS/DMP_Multscale96.ps.Z>.Acesso em: nov.2000.

DORNELES, R. V. **Particionamento de Domínio e Balanceamento Dinâmico de Carga em Arquiteturas Heterogêneas**: aplicação a modelos hidrodinâmicos e de transporte de massa 2d e 3d. Porto Alegre: Universidade Federal do Rio Grande do Sul, 2001. Proposta de Tese.

DORNELES, R. V.; RIZZI, R. L.; DIVERIO, T. A.; NAVAUX, P. O. A. A h-Refined Schwarz-Krylov Solution for Hydrodynamics and Mass Transport in a PCs Cluster. In: SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING, SBAC-PAD, 13., 2001, Pirenópolis, GO. **Proceedings...** Brasilia: UnB, 2001.

DORNELES, R. V.; RIZZI, R. L.; ZEFERINO, C. A.; DIVERIO, T. A.; BAMPI, S.; SUZIN, A. A.; NAVAUX, P. O. A. Parallel Solution for Shallow Water Equations Using Data Decomposition. In: JORNADAS CHILENAS DE COMPUTACION, 2000, Santiago, Chile. **Anales...** Santiago:Universidad de Santiago, 2000. 1 CD-ROM.

DORNELES, R. V.; RIZZI, R. L.; ZEFERINO, C. A.; DIVERIO, T. A.; NAVAUX, P. O. A.; BAMPI, S.; SUZIN, A. A. A PC Cluster Implementation of a Mass Transport Two Dimensional Model. In: SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING, SBAC-PAD, 12., 2000, São Pedro, SP. **Proceedings...** São Carlos: UFSCAR, 2000. p.17–24.

DRIESSCHE, R. V.; ROOSE, D. An Improved Spectral Bisection Algorithm and its Application to Dynamic Load Balancing. **Parallel Computing**, [S.l.], v.21, n.10, p.29–48, Jan. 1995.

FARHAT, C. A Simple and Efficient Automatic Fem Domain Decomposer. **Computers and Structures**, [S.l.], v.28, n.5, p.579–602, 1988.

FERENCZ, A.; SZEWCZYK, R.; WEINSTEIN, J.; WILKENING, J. **Graph Bisection. CS 270 - Final Report**. Disponível em: <http://www.cs.berkeley.edu_szewczyk/CS270>. Acesso em: jan. 2001.

FIDUCCIA, C. M.; MATTHEYSES, M. A Linear-time Heuristic for Improving Network Partition. In: IEEE DESIGN AUTOMATION CONFERENCE, 19., 1982. **Proceedings...** New York:ACM/IEEE, 1982. p.175–182.

FOSTER, I. T. **Designing and Building Parallel Programs**. Reading, USA: Addison-Wesley, 1994. 381p.

FOX, G. C.; WILLIAMS, R. D.; MESSINA, P. C. **Parallel Computing Works!** [S.l.]: Morgan Kaufmann Publishers, 1994.

GALANTE, G.; BALBINOT, J.; RIZZI, R. L.; DORNELES, R. V.; DIVERIO, T. A. Processamento de Alto Desempenho Aplicado a Modelos Computacionais de Dinâmica de Fluidos Ambiental. In: ESCOLA REGIONAL DE ALTO DESEMPENHO, ERAD, 2., 2002, São Leopoldo, RS. **Anais...** Porto Alegre:SBC, 2002. p.271–274.

GALIL, Z. Efficient Algorithms for Finding Maximum Matching in Graphs. **Computing Surveys**, New York, v.18, n.1, p.23–38, Mar. 1986.

GAREY, M. R.; JOHNSON, D. S. **Computers and Intractability** : a guide to the theory of np-completeness. 20th ed. New York:W.H.Freeman,1999. 338p.

GHOSH, G.; MUTHUKRISHNAN, U.; SCHULTZ, M. First and Second Order Diffusive Methods for Rapid, Coarse, Distributed Load Balancing. In: ANNUAL ACM SYMPOSIUM ON PARALLEL ALGORITHMS AND ARCHITECTURES, 8., 1996, Padua, Italia. **Proceedings...** [S.l.: s.n.], 1996. p.72–81.

GILBERT, J. R.; ZMIJEWSKI, E. A Parallel Graph Partitioning Algorithm for a Message-Passing Multiprocessor. **International Journal of Parallel Programming**, New York, v.16, p.498–513, 1987.

GOLUB, G. H.; LOAN, C. F. V. **Matrix Computations**. Baltimore, USA: John Hopkins University Press, 1996. 694p.

GROPP, W.; LUSK, E. **Installation Guide to Mpich, a Portable Implementation of MPI Version 1.2.1**. Disponível em: <http://www.mcs.anl.gov/mpi/mpich/download>. Acesso em: set. 2000.

GROPP, W.; LUSK, E.; DOSS, N.; SKJELLUM, A. A High-performance, Portable Implementation of the MPI Message Passing Interface Standard. **Parallel Computing**, Netherlands, v.22, p.789–828, Sept. 1996.

HENDRICKSON, B.; KOLDA, T. Partitioning Rectangular and Structurally Nonsymmetric Sparse Matrices for Parallel Processing. **SIAM Journal on Scientific Computing**, Philadelphia, v.21, n.6, p.2048–2072, Jan. 1998.

HENDRICKSON, B.; LELAND, R. **Multidimensional Spectral Load Balancing**. Disponível em: <http://www.cs.sandia.gov/~bahendr/papers.html>. Acesso em: out. 2000.

HENDRICKSON, B.; LELAND, R. **The Chaco User´s guide. Version 2.0**. Disponível em: <http://www.cs.sandia.gov/CRF/papers_chaco.html>. Acesso em: set. 2000.

HOUSTIS, C. E. **Load Balancing for Grid/mesh Based Problems in Computational Science and Engineering**. Disponível em: <http://www. ics.forth.gr/ PROJ/ PLEIADES/PROJECTS/ LYDIA/1st-workshop/ Houstis/Houstis.html>. Acesso em: jan. 2001.

HU, Y. F.; BLAKE, R. J. An Optimal Migration Algorithm for Dynamic Load Balancing. **Concurrency: Practice and Experience**, [S.l.], v.10, p.467–483, 1998.

HU, Y. F.; BLAKE, R. J. An Improved Diffusion Algorithm for Dynamic Load Balancing. **Parallel Computing**, Netherlands, v.25, p.417–444, 1999.

KARYPIS, G.; KUMAR, V. **A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs**. Disponível em: <http://www.cs.umn.edu/~karypis>. Acesso em: maio 2002.

KARYPIS, G.; KUMAR, V. **METIS**: a software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. Disponível em: <http://www.cs.umn.edu/~karypis>. Acesso em: maio 2002.

KARYPIS, G.; KUMAR, V. Multilevel K-way Partitioning Schemes for Irregular Graphs. **Journal of Parallel and Distributed Computing**, Orlando, v.48, n.1, p.96–129, Jan. 1998.

KARYPIS, G.; KUMAR, V. Multilevel Algorithms for Multi-Constraint Graph Partitioning. In: SUPERCOMPUTING, 1998. **Proceedings...** [S.l.: s.n.], 1998.

KERNIGHAN, B.; LIN, S. An Effective Heuristic Procedure for Partitioning Graphs. **The Bell System Technical Journal**, New York, p.291–308, Feb. 1970.

KWOK, K.; LAM, F.; HAMDI, M.; PAN, Y.; HUI, C. Application-Specific Load Balancing on Heterogeneous Systems. In: BUYYA, R. (Ed.). **High Performance Cluster Computing. Vol.2 Programming and Applications**. New Jersey, USA: Prentice Hall, 1999. p.350–374.

LUCAS, B. **CS267 - Applications of Parallel Computers. Lecture 14**: graph partitioning ii. Disponível em: <http://www.nersc.gov/~dhbailey/cs267>. Acesso em: jan. 2001.

MARTINOTTO, A. L. **Paralelização de Métodos Numéricos de Resolução de Sistemas Esparsos de Equações Utilizando MPI e de Pthreads**. 2001. Trabalho de Conclusão (Curso de Ciência da Computação) — Universidade de Caxias do Sul, Caxias do Sul, RS.

MESSINGER, F. **Numerical Methods**: the arakawa approach, horizontal grid, global and limited-area modeling. Camp Springs, USA: Academic Press, 1998.

PALHA, M. A. K. **Avaliação de Desempenho e Perfilamento de Código em Programas Paralelos**. 2000. Trabalho de Conclusão (Curso de Ciência da Computação) — Universidade de Caxias do Sul, Caxias do Sul, RS.

PARASHAR, M.; BROWNE, J. C. Distributed Dynamic Data-Structures for Parallel Adaptive Mesh Refinement. In: INTERNATIONAL CONFERENCE OF HIGH PERFORMANCE COMPUTING, 1995. **Proceedings...** [S.l.: s.n.], 1995.

PELLEGRINI, F. **Scotch 3.1 User's Guide**. Disponível em: <http://dept-info.labri.u-bordeaux.fr/~pelegrin/scotch>. Acesso em: jan. 2001.

PICININ, D. J. **Paralelização do Algoritmo do Gradiente Conjugado Através da Biblioteca MPI e de Threads**. 2001. Trabalho Individual (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

PICININ, D. J. **Paralelizações de Métodos Numéricos em Clusters Empregando as Bibliotecas**: mpich, deck e pthreads. 2003. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre.

PICININ, D. J.; MARTINOTTO, A. L.; RIZZI, R. L.; DORNELES, R. V.; DIVERIO, T. A.; NAVAUX, P. O. Ordenação de Mensagens e Pré-Condicionamento na Solução Paralela do Gradiente Conjugado em Clusters de PCs Multiprocessados. In: WORKSHOP EM SISTEMAS COMPUTACIONAIS DE ALTO DESEMPENHO, WSCAD, 3., 2002, Vitória, ES. **Anais...** Vitória:UFES, 2002. p.95–102.

PICININ, D. J.; RIZZI, R. L.; DORNELES, R. V.; DIVERIO, T. A. Paralelização de Métodos Iterativos do Subespaço de Krylov em Clusters Multiprocessados. In: CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL, CNMAC, 24., 2001, Belo Horizonte, MG. **Resumo das Comunicações...** Belo Horizonte:UNI-BH, 2001. p.325.

PICININ, D. J.; RIZZI, R. L.; DORNELES, R. V.; DIVERIO, T. A. Parallelizing Conjugate Gradient Method for Clusters using MPI and Threads. In: INTERNATIONAL CONFERENCE ON PARALLEL AND DISTRIBUTED PROCESSING TECHNIQUES AND APPLICATIONS, PDPTA, 2002, Las Vegas. **Proceedings...** [Las Vegas]:CSREA, 2002.

PILKINGTON, J. R. Dynamic Partitioning of Non-Uniform Structured Workloads with Spacefilling Curves. **IEEE Transactions on Parallel and Distributed Systems**, New York, v.7, n.3, p.228–300, Mar. 1996.

POVITSKY, A. **Parallelization of the Pipelined Thomas Algorithm**. Hampton: Institute for Computer Application in Science and Engineering, 1998. (ICASE REPORT n. 98-48).

QUINN, M. J. **Parallel Computing** : theory and practice. New York: McGraw-Hill, 1994. 446p.

RIZZI, R. L. **Modelo Computacional Paralelo para a Hidrodinâmica e para o Transporte de Massa Bidimensional e Tridimensional**. 2002. 254p. Tese (Doutorado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

RIZZI, R. L.; DORNELES, R. V.; CANAL, A. P.; GOULART, P. C.; DIVERIO, T. A. Modelos Computacionais 2-D e 3-D para a Hidrodinâmica e Transporte de Massa. In: ESCOLA REGIONAL DE ALTO DESEMPENHO, ERAD, 1., 2001, Gramado, RS. **Anais...** Porto Alegre:SBC, 2001. p.226–228.

RIZZI, R. L.; DORNELES, R. V.; DIVERIO, T. A. Solução da Equação do Transporte Bidimensional usando a Técnica ADI. In: CONGRESSO NACIONAL DE MATEMÁTICA APLICADA E COMPUTACIONAL, CNMAC, 22., 1999, Santos, SP. **Resumo das Comunicações...** [S.l.:SBMAC], 1999. p.86–87.

RIZZI, R. L.; DORNELES, R. V.; DIVERIO, T. A. **Modelo Computacional Paralelo para a Hidrodinâmica e para o Transporte de Massa Bidimensional e Tridimensional**. Porto Alegre: Universidade Federal do Rio Grande do Sul, 2003. Relatório de Pesquisa (a publicar).

RIZZI, R. L.; DORNELES, R. V.; NAVAUX, P. O. A.; DIVERIO, T. A. Parallel Solution in PC Clusters by the Schwarz Domain Decomposition for Three-dimensional Hydrodynamics Models. In: INTERNATIONAL MEETING ON HIGH PERFORMANCE COMPUTING FOR COMPUTATIONAL SCIENCE, VECPAR, 5., 2002, Porto, Portugal. **Proceedings...** Porto:Faculdade de Engenharia da Universidade do Porto, 2002. p.655–667.

RIZZI, R. L.; ZEFERINO, C. A.; DORNELES, R. V.; NAVAUX, P. O. A.; BAMPI, S.; SUZIN, A. A.; DIVERIO, T. A. Fluvial Fflowing of Guaiba River Estuary: a parallel solution for the shallow water equations model. In: INTERNATIONAL MEETING ON HIGH PERFORMANCE COMPUTING FOR COMPUTATIONAL SCIENCE, VECPAR, 4., 2000, Porto, Portugal. **Proceedings...** Porto:Faculdade de Engenharia da Universidade do Porto, 2000. p.885–896.

ROEST, M. R. T. **Partitioning for Parallel Finite Diff erence Computations in Coastal Water Simulations**. 1997. Tese (Doutorado em Ciência da Computação) — Delft University of Technology, Delft, Holanda.

SAAD, Y. **Iterative Methods for Sparse Linear Systems**. Boston: PWS Publishing Company, 1996. 447p.

SANTOS, L. L. P. dos. **Particionamento de Grafos**. 2001. Trabalho de Conclusão (Curso de Ciência da Computação) — Universidade de Caxias do Sul, Caxias do Sul, RS.

SCHLOEGEL, K.; KARYPIS, G.; KUMAR, V. **A Performance Study of Ddiffusive vs. Remapped Load-balancing Schemes**. Disponível em: <http://www-users.cs.umn.edu/~karypis/publications/ partitioning.html>. Acesso em: nov. 2000.

SCHLOEGEL, K.; KARYPIS, G.; KUMAR, V. **Dynamic Repartitioning of Adaptively Refined Meshes**. Disponível em: <http://www.arc.umn.edu/publications/preprint/ abstracts99.html#99-006>. Acesso em: nov. 2000.

SCHLOEGEL, K.; KARYPIS, G.; KUMAR, V. **Multilevel Repartitioning of Adaptive Meshes**. Disponível em: <http://www-users.cs.umn.edu/~karypis/ publications/Talks/adaptive>. Acesso em: nov. 2000.

SHEWCHUCK, J. R. **An Introduction to the Conjugate Gradient Method Without the Agonizing Pain**. [S.l.]: Carnegie Mellon University, 1994. Disponível em: <ftp://WARP.CS.CMU>.EDU. Acesso em: out. 1999.

SHU, W.; WU, M.-Y. An Incremental Parallel Scheduling Approach to Solving Dynamic and Irregular Problems. In: INTERNATIONAL CONFERENCE ON PARALLEL PROCESSING, 24., 1995, Oconomowoc, WI. **Proceedings...** [S.l.: s.n.], 1995. v.2, p.143–150.

SIMON, H. D. **Partitioning of Unstructured Problems for Parallel Processing**. [S.l.]: NASA Ames Research Center, 1991. (RNR-91-008).Disponível em: <http://www.nas.nasa.gov/Research/Reports/Techreports/ 1991/rnr-91-008-abstract.html>. Acesso em: jan. 2001.

SIMON, H. D.; TENG, S. **How good is recursive bisection**. [S.l.]: NASA Ames Research Center, 1993. (RNR-93-012).Disponível em: <http://www.nas.nasa.gov/Research/Reports/Techreports/ 1993/rnr-93-012-abstract.html>. Acesso em:jan. 2001.

SJöSTROM, E. **Singular Value Computations for Toeplitz Matrices**. Disponível em: <http://www.math.liu.se/_evlun/pub/lic/lic.html>. Acesso em: nov. 2000.

VOLLEBREGT, E.; M.R.T.ROEST; CATE, H.; H.X.LIN. Performance Analysis of Parallel TRIWAQ I: cluster of workstations and parsytecxplores. In: MMARIE ANNUAL WORKSHOP, 2., 1997, Barcelona, Espanha. **Proceedings...** [S.l.: s.n.], 1997. p.1–18.

WALSHAW, C. **The Jostle User Manual**: version 2.2. Disponível em: <http://www.gre.ac.uk/jostle>. Acesso em: maio 2001.

WALSHAW, C.; CROSS, M.; MCMANUS, K. **Multiphase Mesh Partitioning**. London, UK: Universidade de Greenwich, 1999. (99/IM/51).

WATTS, J. **A Practical Approach to Dynamic Load Balancing**. 1995. 47p. Dissertação (Mestrado em Ciência da Computação) — Scalable Concurrent Programming Laboratory, California Institute of Technology, California, USA.

WEBBER, R. F. **Arquitetura de Computadores Pessoais**. Porto Alegre, RS: Sagra-Luzzatto, 2000.

WILLEBEEK-LEMAIR, M.; REEVES, A. P. Strategies for Dynamic Load Balancing on Highly Parallel Computers. **IEEE Transactions on Parallel and Distributed Systems**, New York, v.4, n.9, p.979–993, 1993.

XU, C.; LAU, F. Optimal Parameters for Load Balancing with the Diffusion Method in Mesh Networks. **Parallel Processing Letters**, [S.l.], v.4, n.2, p.139–147, June 1994.

ZAGHA, M.; LARSON, B.; TURNER, S.; ITZKOWITZ, M. Performance Analysis Using the MIPS R10000 Performance Counters. In: SUPERCOMPUTING, 1996, Pittsburgh, USA. **Proceedings...** [S.l.: s.n.], 1996.

ZEFERINO, C. A.; RIZZI, R. L.; DORNELES, R. V.; BAMPI, S.; SUZIN, A.; DIVERIO, T. A. A Parallel Simulation of the Hydrodynamics of Guaiba River. In: UFRGS MICROELECTRONICS SEMINAR, SIM, 15., 2000, Torres, RS. **Proceedings...** Porto Alegre:Instituto de Informática da UFRGS, 2000. p.21–24.

ZNATI, T. F.; MELHEM, R. G.; PRUHS, K. R. Dilation Based Bidding Schemes for Dynamic Load Balancing on Distributed Processing Systems. In: DISTRIBUTED MEMORY COMPUTING CONFERENCE, 6., 1991. **Proceedings...** Portland:IEEE Computer Society Press, 1991. p.129–136.

ZUMBUSCH, G. Load Balancing for Adaptively Refined Grids. In: APPLIED MATHEMATICS AND MECHANICS, 2001, Zurique, Suiça. **Proceedings...** [S.l.: s.n.], 2001.