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

CLAUDIO SCHEPKE

# Distribuição de Dados para Implementações Paralelas do Método de Lattice Boltzmann

Dissertação apresentada como requisito parcial
para a obtenção do grau de
Mestre em Ciência da Computação

Prof. Dr. Tiarajú Asmuz Diverio
Orientador

Prof. Dr. Nicolas Bruno Maillard
Co-orientador

Porto Alegre, março de 2007

**CIP – CATALOGAÇÃO NA PUBLICAÇÃO**

Schepke, Claudio

Distribuição de Dados para Implementações Paralelas do Método de Lattice Boltzmann / Claudio Schepke. – Porto Alegre: PPGC da UFRGS, 2007.

91 f.: il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação, Porto Alegre, BR–RS, 2007. Orientador: Tiarajú Asmuz Diverio; Co-orientador: Nicolas Bruno Maillard.

1. Lattice Boltzmann. 2. Particionamento de Dados em Bloco. 3. Processamento Paralelo. 4. Dinâmica de Fluidos Computacional. 5. Aplicações de Alto Desempenho. 6. Simulação Numérica. 7. Computação Científica. I. Diverio, Tiarajú Asmuz. II. Maillard, Nicolas Bruno. III. Título.

*"As pessoas podem fazer seus planos,*
*porém é o SENHOR Deus quem dá a última palavra,*
*O coração do homem pode fazer planos,*
*mas a resposta dos lábios vem do Senhor"*
— Pv 16.1 RA

# AGRADECIMENTOS

Reconheço a participação de Deus em toda a minha vida. É Ele quem dirige todos os meus passos. Sem Ele eu não teria conseguido superar mais essa etapa da minha vida. Não um deus qualquer, mas um Deus que me ama e que está próximo de mim através da fé implantada por Ele em meu coração. Uma fé que aceita o perdão que Jesus Cristo me oferece, a qual garante uma vida eterna no céu.

Agradeço também:

Aos meus pais por sempre estarem preocupados comigo e com o andamento dos meus trabalhos. Por eles terem me proporcionado condições de chegar até aqui.

A Diana, minha namorada, que me acompanha desde o início de produção desta Dissertação. Por todo seu amor, carinho e compreensão.

Aos meus amigos espalhados pelo Estado, Brasil e Mundo! Durante estes últimos dois anos eu pude fazer muitos amigos fora da minha vida acadêmica, enquanto outros tiveram seus laços de amizade reforçados. Tais pessoas tem sido importantes nas decisões de minha vida.

Ao professor Tiarajú, pela sua amizade, orientação e acolhimento. Um pessoa muito especial que sempre indicou os melhores caminhos, bem como resolveu eficientemente questões administrativas, tornando minha vida de estudante mais fácil. Ótimo companheiro de viagem, sendo também bastante compreensivo em relação a todas as questões da vida.

Ao professor Nicolas por ter norteado a etapa final do meu trabalho.

Aos colegas em seus mais diversos níveis de formação, os quais, em sua maioria, são meus companheiros de pesquisa oriundos do Laboratório de Sistemas de Computação (LSC) da Universidade Federal de Santa Maria (UFSM). Também aos demais estudantes, vindo de diferentes universidades, que compuseram e compõem o Grupo de Processamento Paralelo e Distribuído (GPPD).

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro.

Ah, sem esquecer, a todos os administradores de clusters por manterem a infraestrutura das máquinas funcionando para os testes...

Obrigado!

# SUMÁRIO

# LISTA DE ABREVIATURAS E SIGLAS

| | |
|---|---|
| 2D | Bidimensional |
| 3D | Tridimensional |
| BGK | Bhatnagar, Gross e Krook |
| RDD | *Regular Domain Decomposition* |
| DF | Dinâmica de Fluidos |
| DFC | Dinâmica de Fluidos Computacional |
| EB | Equação de Boltzmann |
| EDP | Equação Diferencial Parcial |
| EE | Equação de Euler |
| ELB | Equação Lattice Boltzmann |
| ENS | Equações de Navier-Stokes |
| FHP | Frisch, Hasslacher e Pomeau |
| GPPD | Grupo de Processamento Paralelo e Distribuído |
| HPP | Hardy, Pazzis e Pomeau |
| II | Instituto de Informática |
| LGA | Lattice Gas Automata |
| LogP | *Latency, overhead, gap and Processors* |
| MLB | Método de Lattice Boltzmann |
| MPI | *Message Passing Interface* |
| ORB | *Orthogonal Recursive Bisection* |
| SPMD | *Single Program Multiple Data* |
| UFRGS | Universidade Federal do Rio Grande do Sul |

# LISTA DE SÍMBOLOS

| | |
|---|---|
| $\frac{\partial}{\partial x}$ | [-] Derivada parcial em relação a um termo $x$ |
| $\epsilon$ | [$J$] Energia interna |
| $\varepsilon$ | [-] Número de Knudsen |
| $\lambda$ | [-] Tempo de relaxação adimensional |
| $\mu$ | [$m^2/s$] Viscosidade cinética |
| $\rho$ | [$kg/m^3$] Densidade |
| $\tau$ | [$s$] Tempo de relaxação |
| $\Omega$ | [-] Operador de colisão/relaxação |
| $\Pi_{\alpha\beta}$ | [-] Tensor de fluxo de momento |
| $\nabla$ | [-] Gradiente: soma dos vetores de derivadas parciais |
| $\nabla^2$ | [-] Operador Laplaciano |
| $a$ | [$m/s^2$] Aceleração |
| $c$ | [$m/s$] Velocidade do som |
| $c_s$ | [$m/x$] Velocidade do som no MLB, equivalente a $c/\sqrt{3}$ |
| $d$ | [-] Número de direções possíveis de uma partícula em um reticulado |
| $e_i$ | [$m/s$] Vetor de velocidade no reticulado |
| $E$ | [$J$] Energia total |
| $f$ | [$kg/m^3$] Função de distribuição das micro-partículas |
| $\overline{f}$ | [$kg/m^3$] Função de distribuição de equilíbrio de Maxwell-Boltzmann |
| $F$ | [$kgm/s$] Força externa |
| $g$ | [s] Intervalo mínimo entre o envio/recebimento de mensagens |
| $i$ | [-] Índice do número de direções de deslocamento |
| $j$ | [-] Índice de elementos do reticulado |
| $k_B$ | [$J/k$] Constante de Boltzmann |
| $L$ | [$s$] Latência de Rede |
| $m$ | [$kg$] Massa |

| | |
|---|---|
| $n(x,t)$ | [-] Distribuição das partículas no reticulado em LGA |
| $N$ | [-] Número de partículas |
| $p$ | [$kg/ms^2$] Pressão |
| $Q$ | [$J/s$] Transferência de calor |
| $s$ | [$J/K$] Entropia |
| $t$ | [$s$] Tempo |
| $\Delta t$ | [s] Variação de tempo |
| $T$ | [$K$] Temperatura |
| $v$ | [$m/s$] Velocidade |
| $v_{out}$ | [$bit/s$] Vazão |
| $w_i$ | [-] Fator usado na função de equilíbrio |
| $x$ | [$m$] Posição espacial |
| $\Delta x$ | [$m$] Variação de espaço |

# LISTA DE FIGURAS

# LISTA DE TABELAS

# RESUMO

A Dinâmica de Fluidos Computacional é uma importante área de pesquisa no contexto da Computação Científica. Através da modelagem e simulação das propriedades de líquidos e gases é possível obter resultados numéricos para diferentes estruturas e fenômenos físicos cotidianos e de grande importância econômica. A evolução dos sistemas computacionais possibilitou a essa área o surgimento de novas técnicas e abordagens de simulação. Uma das técnicas computacionais atualmente empregadas é o Método de Lattice Boltzmann, um método numérico iterativo para a modelagem e simulação mesoscópica da dinâmica de fluxos de fluidos. Diferentes tipos de sistemas físicos podem ser tratados através dessa técnica, como é o caso de fluxos em meios porosos ou de substâncias imiscíveis. No entanto, por causa da dimensão dos sistemas físicos, é necessário adotar estratégias que permitam a obtenção de resultados precisos ou em tempos computacionais aceitáveis. Assim, paralelizar as operações é a solução mais indicada para aumentar o desempenho do método. Uma maneira eficiente de paralelizar um método numérico é fazer uso de técnicas de distribuição de dados refinadas, como é o caso do particionamento em blocos. Tais abordagens de paralelização foram adotadas neste trabalho em implementações bi- e tridimensionais do Método de Lattice Boltzmann, com o intuito de avaliar o ganho de desempenho oferecido através dessa técnica. Além disso, foram definidos os fatores que influenciam as melhores configurações de particionamento. Os resultados obtidos demonstraram que o particionamento em blocos prove um aumento considerável do desempenho das aplicações paralelas, especialmente para a versão tridimensional do método. Para algumas configurações dos estudos de caso os tempos de execução diminuíram em até 30% em relação aos tempos obtidos com o particionamento unidimensional. Já as melhores configurações para a distribuição dos dados em blocos foram aquelas em que a disposição dos dados manteve-se mais quadrada ou cúbica em relação a cada uma das dimensões coordenadas.

**Palavras-chave:** Lattice Boltzmann, Particionamento de Dados em Bloco, Processamento Paralelo, Dinâmica de Fluidos Computacional, Aplicações de Alto Desempenho, Simulação Numérica, Computação Científica.

**Data Distribution for Parallel Implementations of the Lattice Boltzmann Method**

# ABSTRACT

Computational Fluid Dynamics is an important research area in the Scientific Computing context. Through the modeling and simulation of liquids and gases properties it is possible to get numerical results for different physical structures and daily phenomena that have great economic importance. The evolution of the computational systems made it possible to develop new techniques and approaches of simulation in this area. One of these techniques currently used is the Lattice Boltzmann Method. This method is an iterative numerical strategy for modeling and simulating mesoscopic dynamics of fluid flows. Different types of physical systems can be simulated through this technique, like immiscible substances and flows in porous media. However, since the dimension of the physical systems is usually large, it is necessary to adopt strategies that allow to get accurate results or results in an acceptable computational time. Thus, the parallelization of the operations is the best alternative to increase the performance of the method. An efficient way to parallelize a numerical method is to make use of refined data distribution techniques, like data partitioning in blocks. Such parallelization approach had been adopted in this work for bi- and three-dimensional implementations of the Lattice Boltzmann Method. The objective of the work was to evaluate the performance enhancement offered through the parallelization. Moreover, another objective is to define the elements that influence the best partitioning configurations. The results shown that data partitioning in blocks provide a considerable performance increase for parallel implementations, especially for the three-dimensional version of the method. For some configurations adopted in the case studies, the execution time was reduced of up to 30% in relation to the one-dimensional partitioning strategy. The best configurations for data distribution in blocks were that where the data disposal are more square or cubical shaped in relation to each one of the coordinate dimensions.

**Keywords:** Lattice-Boltzmann, Block Data Partitioning, Parallel Processing, Computational Fluid Dynamics, High Performance Application, Numerical Simulation, Scientific Computing.

# 1 INTRODUÇÃO

O uso de arquiteturas paralelas (ANDREWS, 2001) tem sido recorrente na resolução de Aplicações Científicas (LUCQUIN; PIRONNEAU, 1998) e, em especial, na área da Dinâmica de Fluidos Computacional. Esse fato pode ser observado na descrição das principais áreas de pesquisa feitas utilizando as 500 máquinas de maior capacidade de processamento existentes no mundo, as quais se encontram relacionadas na lista *TOP500* (*http://www.top500.org*). Além dessas pesquisas, existem ainda, em nível local, diversos trabalhos desenvolvidos pelo Grupo de Processamento Paralelo e Distribuído (GPPD) do Instituto de Informática (II) da Universidade Federal do Rio Grande do Sul (UFRGS) que exploram o paralelismo como forma de aumentar o desempenho de aplicações da Dinâmica de Fluidos Computacional (RIZZI, 2002; DORNELES, 2003; GALANTE, 2006).

## 1.1 Dinâmica de Fluidos Computacional

A Dinâmica de Fluidos (DF) é uma área de pesquisa de grande relevância tecnológica. Através do estudo de fenômenos que ocorrem com líquidos e gases a DF busca relacionar diferentes propriedades físicas, tais como: velocidade, pressão, temperatura e densidade em função do tempo e do espaço (BATCHELOR, 1987). Os trabalhos desenvolvidos nessa área possibilitam a simulação de diversos sistemas físicos associados a fenômenos cotidianos e de grande relevância, incluindo o efeito de maremotos, a simulação de furacões, a previsão de secas ou inundações e a distribuição de partículas poluentes na atmosfera ou em reservatórios naturais de água.

A evolução dos sistemas computacionais disponibilizou à DF novos rumos as pesquisas e, conseqüentemente, o avanço da Ciência. Com isso, foi possível melhorar os métodos de resolução baseados em simulações numéricas, modelar diversos fenômenos e estruturas, visualizar sistemas físicos através de modelos específicos (SIMS et al., 2000), bem como explorar o paralelismo com a finalidade de obter resultados de forma mais eficiente. Essa área é conhecida como Dinâmica de Fluidos Computacional (DFC), sendo bastante abrangente no contexto da Computação Científica (ANDERSON, 1995; VERSTEEG; MALALASEKRA, 1995).

Na DFC, fluidos são normalmente descritos através das Equações de Euler (EE) ou de Navier-Stokes (ENS) (LANDAU; LIFSHITZ, 1982). Tais equações relacionam as diferentes propriedades físicas e as forças que governam um determinado fluxo macroscópico. Uma vez que a solução algébrica não é viável, resolver computacionalmente o sistema de equações passa a ser uma das melhores alternativas encontradas. Para que isso seja possível, é necessário, primeiramente, que o sistema de equações seja discretizado. A partir da representação discreta e das definições das condições de contorno, pode-se obter a solução apropriada para as equações através de algum método numérico específico.

Embora a simulação em nível macroscópico seja perfeitamente viável, apenas em uma pequena parcela das simulações é possível gerar resultados suficientemente precisos (BUICK, 1997). Isso porque abordagens macroscópicas são geralmente insensíveis à dinâmica microscópica envolvida, de modo que as quantidades físicas que deveriam ser modificadas dinamicamente durante a simulação acabam permanecendo constantes (WOLF; SANTOS; PHILIPPI, 2006). Conseqüentemente, as soluções de todas as equações que descrevem um fluxo somente podem ser feitas de forma aproximada.

Uma outra abordagem possível para a simulação de fluidos é a utilização de um modelo molecular (HOCKNEY; EASTWOOD, 1988). Nesse caso, a simulação ocorre em nível de interação entre moléculas, as quais são regidas pelas leis de movimento de Newton, que determinam a posição e a velocidade de cada molécula (BHATNAGAR; GROSS; KROOK, 1954). Já as propriedades físicas são obtidas através de resultados estatísticos do comportamento coletivo das moléculas. Do ponto de vista computacional, essa abordagem ainda possui limitações, visto que as escalas de espaço e tempo possíveis de serem simuladas estão longe do tamanho real de muitas aplicações (WOLF; SANTOS; PHILIPPI, 2006).

Observando a importância da dinâmica microscópica e ao mesmo tempo superando as limitações do modelo molecular, foram introduzidos modelos mesoscópicos de sistemas de partículas (HARDY; POMEAU; PAZZIS, 1973). Um dos primeiros modelos desenvolvidos foi a técnica de Lattice Gas Automata (LGA) (FRISCH et al., 1987). Em LGA, as partículas representam um conjunto de moléculas cuja quantidade é inferior a menor escala usada na simulação, o que reduz o número de pontos a ser simulado. A simulação ocorre em tempo, espaço e velocidades discretas, tornando simples sua implementação. Além disso, a interação entre as partículas ocorre apenas entre os elementos vizinhos através de operações lógicas.

Apesar da praticidade do modelo LGA, existem algumas limitações na modelagem das propriedades físicas dos fluidos, especialmente na forma de representá-las. As modificações realizadas em LGA deram origem ao Método de Lattice Boltzmann (MLB) (CHEN; DOOLEN, 1998; SUCCI, 2001). Diferente de LGA, o MLB não considera o movimento individual das partículas, mas usa valores reais, ao invés de lógicos, para a propagação das informações pelo reticulado. Além de ser entendido como uma evolução de LGA, o MLB pode ser visto também como uma representação discreta da Equação de Boltzmann (EB), base da teoria cinética dos gases (CHEN; DOOLEN, 1998; HE; LUO, 1997a; MCNAMARA; ZANETTI, 1988). O método satisfaz, ainda, as ENS para dinâmica de fluidos em regime incompressível (HE; LUO, 1997b).

As aplicações práticas do MLB possibilitam a modelagem computacional de uma ampla variedade de problemas, incluindo a simulação de escoamentos de fluidos em estruturas complexas e escoamentos de fluidos imiscíveis. A literatura apresenta muitas aplicações que utilizam implementações do MLB. Isso pode ser visto em trabalhos relacionados à hemodinâmica (FANG et al., 2002), fenômenos gasosos (WEI et al., 2004), polímeros líquidos (ONISHI; CHEN; OHASHI, 2005), nanofluidos (XUAN; YU; LI, 2005) e transporte de substâncias (MASSELOT, 2000). Também existem alguns trabalhos envolvendo a simulação de fluxos em geometrias complexas, especialmente em meios porosos como rochas, concreto e solo (WU et al., 2005; GUOI; ZHAO, 2005).

Em termos de simulação computacional, o MLB têm uma necessidade maior de recursos do que os exigidos por outros métodos numéricos, como é o caso das técnicas discretas de resolução das ENS (SUCCI, 2001). Devido à relevância do MLB e da dimensão dos problemas que podem ser explorados através dessa técnica, os quais atingem tempos

computacionais elevados ou inaceitáveis, é importante que o método apresente bons níveis de desempenho. Como as interações são essencialmente locais, envolvendo apenas as partículas vizinhas, paralelizar as operações apresenta-se como uma ótima possibilidade para agregar eficiência ao método (KÖRNER et al., 2005; DUPUIS, 2002). Além disso, a paralelização torna viável a solução de casos onde a quantidade de memória necessária para o armazenamento das informações do reticulado é maior do que o disponível em um sistema monoprocessado. Para tanto, existem diversas alternativas disponíveis, tanto em termos de arquiteturas paralelas, quanto em termos de recursos de programação paralela.

## 1.2 Cálculo Paralelo

Para o desenvolvimento de aplicações paralelas existem basicamente duas abordagens, a saber: o Paralelismo de Tarefas e o Paralelismo de Dados (FOSTER, 1995). No Paralelismo de Tarefas ou Decomposição Funcional a aplicação é particionada em várias tarefas colaborativas. Assim, é possível executar diferentes trechos de código simultaneamente. Já no Paralelismo de Dados ou Decomposição de Domínios os dados são divididos e atribuídos aos processadores, de maneira que cada processador atue somente sobre o seu conjunto de dados. Para tanto, é necessário apenas um único fluxo de instrução.

No caso de aplicações numéricas, tais como as desenvolvidas a partir do MLB, as implementações paralelas são geralmente elaboradas utilizando o Paralelismo de Dados. Um aspecto importante nessa abordagem é a forma de particionar os dados. Uma boa distribuição para arquiteturas homogêneas ocorre quando todos os processadores têm a mesma carga de trabalho. No caso de todas as operações aplicadas sobre os elementos do conjunto de dados serem as mesmas e considerando que a capacidade de processamento dos processadores é semelhante entre si, a melhor forma de distribuir os dados é particionar o problema pelo número total de processadores disponível. No entanto, a maneira como isso ocorre pode influenciar no desempenho das aplicações devido a uma série de fatores tais como: distribuição e acesso aos dados em memória, taxa de acerto de *cache*, dependência de dados e equilíbrio de carga.

Uma maneira de aumentar o desempenho das aplicações paralelas é utilizar uma estratégia de particionamento de dados em bloco. Isso é feito dividindo-se os dados dispostos em estruturas multidimensionais coordenadas em duas ou mais dimensões. O uso de algoritmos paralelos com particionamento em bloco tem como objetivo reduzir a movimentação de dados entre os diversos níveis de hierarquia da memória. Assim, é possível acessar um maior número de regiões de memória contínuas, bem como evitar falhas de acesso a *cache*. Em sistemas de memória distribuída, onde existe dependência de dados, é possível diminuir ainda os custos das comunicações interprocessos.

## 1.3 Objetivos e Contribuição

Sob o contexto apresentado anteriormente, o objetivo desse trabalho consiste em analisar o desempenho paralelo do MLB usando diferentes estratégias de distribuição de dados. Para tanto foram feitas implementações paralelas do método, tanto de um modelo bidimensional, quanto de um modelo tridimensional. Os testes foram feitos em uma arquitetura de *cluster* (WILKINSON; ALLEN, 1998), sendo testadas diferentes configurações de mapeamento de dados aos processadores usando o particionamento em blocos. Com isso, foram determinadas as abordagens de particionamento mais eficientes e os fatores que interagem para a definição dessas melhores formas.

Apesar de existirem avaliações de desempenho para o MLB em diversas arquiteturas, uma lacuna encontrada nos trabalhos relacionados à paralelização do método está na falta de uma análise crítica entre as estratégias de particionamento de dados que podem ser adotadas. Avaliações desse tipo são importantes pois ajudam a definir as estratégias que permitem explorar ao máximo o potencial paralelo de uma arquitetura. Assim, é possível atingir índices mais elevados de desempenho, especialmente em arquiteturas onde isso não ocorre através de esquemas de particionamento mais simples. Neste contexto, a principal contribuição desta dissertação é a resolução paralela de modelos bidimensionais e tridimensionais do MLB, utilizando estratégias otimizadas de particionamento de dados para ambientes de memória distribuída.

## 1.4 Estrutura do Texto

A fim de contextualizar o leitor, o Capítulo 2 e o Capítulo 3 apresentam o embasamento científico deste trabalho, tratando das principais equações da Dinâmica de Fluidos e da fundamentação teórica do MLB, enquanto que o Capítulo 4 relaciona a revisão bibliográfica sobre o tema proposto. Inicialmente, no Capítulo 2 são mostradas as EE e as ENS, que regem as propriedades macroscópicas dos fluidos, e a Equação de Boltzmann, que relaciona a dinâmica microscópica dos fluidos com a dinâmica macroscópica. O Capítulo 3 descreve o MLB, relatando o desenvolvimento do método a partir de LGA, suas equações, os modelos de reticulado, as condições de contorno, o esboço do algoritmo e as áreas de aplicação do método. Já no Capítulo 4 é apresentado o estado da arte das técnicas de paralelização, destacando as implementações paralelas desenvolvidas para o método em trabalhos anteriores.

Os demais capítulos relacionam o desenvolvimento do trabalho propriamente dito. No Capítulo 5 são descritas as implementações paralelas feitas para o MLB, bem como uma análise teórica dos custos de comunicação dos modelos implementados. O Capítulo 6 apresenta os estudos de caso e os resultados obtidos usando o particionamento linear dos dados, além de alguns resultados físicos. Já no Capítulo 7 são destacados e analisados os valores numéricos obtidos através do particionamento em bloco, os quais são comparados com os resultados alcançados com o particionamento linear. Por fim, algumas considerações são apresentadas no Capítulo 8 como conclusão do trabalho.

## 1.5 Siglas, Notação e Terminologia

Como forma de auxiliar o melhor entendimento do texto foram disponibilizados diversos recursos de leitura:

- Os valores por extenso das abreviaturas e siglas utilizadas na dissertação podem ser obtidos na seção LISTA DE ABREVIATURAS E SIGLAS, localizada na Página 7;
- A semântica dos termos (símbolos e letras) usados nas fórmulas é encontrada a partir da Página 8 na seção LISTA DE SÍMBOLOS. Para cada caso é relacionado ainda entre colchetes a unidade de medida que delimita a sua grandeza física. Nos casos onde não existe uma métrica é utilizado um traço como indicador;
- Para a compreensão do significado dos termos técnicos é possível consultar um glossário registrado a partir da Página 88.

# 2 DINÂMICA DE FLUIDOS

A descrição física de um fluido é geralmente feita de forma macroscópica através da Mecânica Contínua, que trata das propriedades físicas de materiais contínuos (sólidos e fluidos), sem se importar com a microestrutura de seus componentes. Nessa abordagem são utilizadas equações diferenciais tanto para expressar as leis fundamentais da física (conservação de massa e momento, por exemplo) quanto para as características específicas do material analisado. As principais equações adotadas para essa abordagem são a Equação de Euler (EE) e a Equação de Navier-Stokes (ENS) (LANDAU; LIFSHITZ, 1982).

Uma forma alternativa para a descrição de um fluido pode ser definido através de um modelo microscópico, usando-se a Física Estatística (REIF, 1965). Uma abordagem estatística apresenta-se adequada para sistemas onde o número de variáveis é muito grande, ao ponto da solução determinística ser impossível de ser encontrada ou irrelevante, ou em dinâmicas não lineares, como é o caso da dinâmica de fluidos. Nesse contexto, os problemas são solucionados através de aproximações e expansões analíticas, ou através de simulações e soluções aproximadas via processamento computacional.

Através da Teoria Cinética dos Gases, desenvolvida por Ludwig Boltzmann no final do século XIX, foram dados os primeiros passos para determinar o comportamento macroscópico de um fluido a partir de um modelo microscópico. No modelo microscópico, o movimento das partículas é descrito através das Leis de Newton, o que possibilitou relacionar as propriedades microscópicas e macroscópicas através de uma função conhecida por Equação de Boltzmann (EB) (CERCIGNANI, 1994; BARDOS; UKAI, 1991). A Figura 2.1 ilustra as duas abordagens discutidas anteriormente, relacionando entre si os modelos e as propriedades físicas.

Além das abordagens microscópica e macroscópica, é possível representar um fluido, ainda, através de uma abordagem mesoscópica. Tal abordagem está baseada em uma escala onde é possível analisar as propriedades do material ou fenômeno, sem se importar com a estrutura individual das partículas. Neste caso, as grandezas médias são suficientes para caracterizar os efeitos observados nos sistemas ali considerados, sem exigir o conhecimento das distribuições das grandezas microscópicas. Modelos de Lattice Gas e de Lattice Boltzmann são exemplos típicos para a representação de fluidos utilizando abordagens mesoscópicas. Uma relação entre esses modelos e as abordagens microscópicas é apresentada na Figura 2.2, onde à medida que a hierarquia se dirige ao topo, diminui a dependência dos detalhes microscópicos.

Sob o contexto discutido anteriormente, este capítulo descreve as equações relacionadas ao processo de desenvolvimento do Método de Lattice Boltzmann. Para tanto, são apresentadas inicialmente as equações de Euler e Navier-Stokes, utilizadas na descrição de fluxos sob o ponto de vista macroscópico. Na seqüência, a Equação de Boltzmann

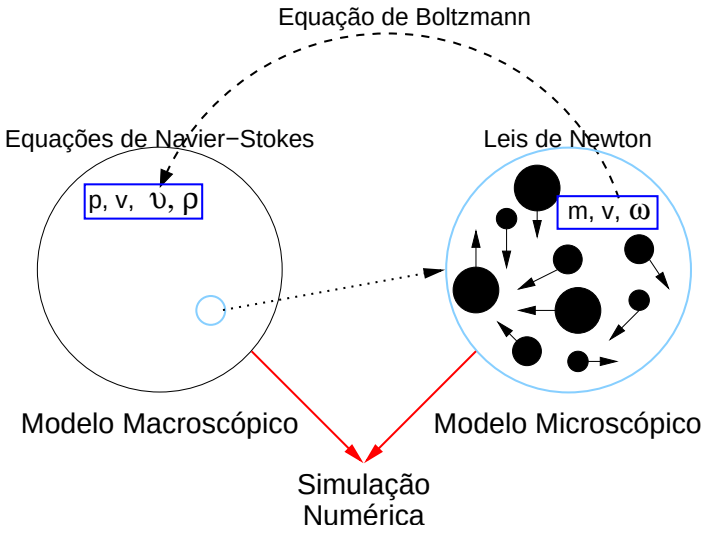


Figura 2.1: Descrição física de um fluido

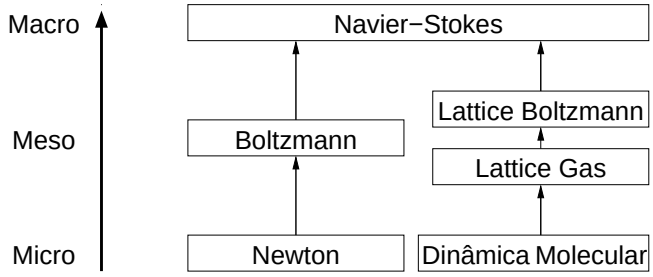


Figura 2.2: Relação entre as abordagens microscópicas, mesoscópicas e macroscópicas

é apresentada, sendo esta a base do desenvolvimento do MLB, o qual será demonstrado posteriormente.

## 2.1 Equações de Euler

Na Dinâmica de Fluidos, as Equações de Euler (EE) governam fluxos onde a viscosidade e a condução do calor não é levada em consideração (LANDAU; LIFSHITZ, 1982). Baseadas na Segunda Lei de Newton, as EE são equivalentes às ENS com viscosidade nula. Sendo $\rho$ a densidade, $t$ o tempo, $v$ a velocidade, $p$ é a pressão e $E$ a energia total dada por unidade de volume (energia cinética e interna), na forma diferencial, as equações de conservação de massa, momento e energia são apresentadas como:

$$\frac{\partial \rho}{\partial t} + \nabla.(\rho v) = 0 \tag{2.1}$$

$$\frac{\partial \rho v}{\partial t} + \nabla.(\rho v)v + \nabla p = 0 \tag{2.2}$$

$$\frac{\partial E}{\partial t} + \nabla.(v(E+p)) = 0, \tag{2.3}$$

Nessas equações, o operador $\nabla$ representa o vetor espacial de derivadas parciais $\left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}\right)$.

## 2.2 Equações de Navier-Stokes

Historicamente, as Equações de Navier-Stokes (ENS) são modificações feitas a partir das EE com o objetivo de inserir as forças de viscosidade ($\mu$) e condução de calor (entropia $s$) oriundas de interações moleculares (LANDAU; LIFSHITZ, 1982). As ENS são um conjunto de Equações Diferenciais Parciais (EDP) não lineares utilizadas para descrever praticamente qualquer fluxo de fluido no qual existe algum interesse de estudo. Tais equações são usadas para representar uma vasta gama de fenômenos, tais como: fluxos de água em canais, correntes oceânicas e movimentos de massas de ar na atmosfera. As soluções das ENS dependem somente das propriedades dos fluidos e das condições de contorno das propriedades físicas analisadas. No entanto, a solução numérica das equações, considerando todos os fatores que interagem na formação das mesmas, é bastante difícil, sendo freqüentemente simplificadas de sua definição original.

Devido às limitações impostas pelo modelo, a solução exata das equações completas somente é possível para os casos em que o fluxo é laminar. Assim, para amostras em pequenas escalas ou sob condições extremas, para fluidos reais baseados em misturas de partículas discretas ou de outros materiais, tais como partículas suspensas e gases dissolvidos, os resultados gerados serão diferentes dos apresentados pela modelagem contínua e homogênea (simulação real). Nesses casos, as ENS descrevem os fluxos apenas aproximadamente.

Uma das premissas para que seja possível descrever um fluxo através das ENS é de que as propriedades do fluxo sejam diferenciáveis e contínuas. Considerando $\mu$ a viscosidade do fluido, $F$ uma força externa que atua através da massa do fluido, $s$ a entropia por unidade de massa, $Q$ a transferência de calor e $T$ a temperatura, as condições de conservação de massa, momento e energia podem ser satisfeitas pelas ENS, sendo definidas, respectivamente pelas equações 2.1, 2.4 e 2.5:

$$\frac{\partial v}{\partial t} = -(v.\nabla)v - \frac{1}{\rho}\nabla p + \mu\nabla^2 v + \frac{F}{\rho} \tag{2.4}$$

$$\frac{\partial s}{\partial t} = -v.\nabla s + \frac{Q}{T} \tag{2.5}$$

A equação de conservação de massa em Navier-Stokes é a mesma que é utilizada em Euler. Na Equação 2.4 é definida a conservação de momento linear através dos vetores das equações irrotacionais de Navier-Stokes, as quais relacionam massa, pressão, viscosidade e força externa (gravidade, inércia, ...) de acordo com a lei de Newton para fluidos (LANDAU; LIFSHITZ, 1982). Já a variação da quantidade de energia de um sistema é definida através da Equação 2.5, onde são consideradas a taxa de transferência de calor e a temperatura do fluido. As equações podem ser simplificadas para determinados casos em que algumas condições são constantes, como é o caso de forças externas nulas, ausência de pressão ou viscosidade baixa e, assim, simplificar a resolução das mesmas.

As ENS são equações parciais de segunda ordem que não possuem uma solução analítica conhecida, exceto para um pequeno número de casos especiais. A Dinâmica de Fluidos Computacional busca apresentar técnicas numéricas para solucionar a não linearidade dos termos das ENS ou de equações derivadas destas através de diferentes métodos (FERZIGER; PERIC, 2002). O uso de modelos lineares e microscópicos para a simulação de fluidos é uma alternativa para a solução direta das equações macroscópicas. Todavia, essas técnicas precisam ser condizentes com as ENS apresentadas anteriormente (FRISCH; HASSLACHER; POMEAU, 1986; CHEN; CHEN; MATTHAEUS, 1992).

## 2.3 Equação de Boltzmann

A Equação de Boltzmann (EB) é uma das principais equação da Teoria Cinética, uma área da Física Estatística que trata da dinâmica de processos em desequilíbrio e sua convergência para o equilíbrio termodinâmico (CERCIGNANI, 1994). A equação foi originalmente desenvolvida para um sistema de expansão de gases. Atualmente, ela pode ser utilizada também para uma série de outras áreas da Física Estatística, tais como as que envolvem a dispersão e transporte de partículas em líquidos, o deslocamento de elétrons em semicondutores e as pesquisas com líquidos quânticos. É importante ressaltar que a derivação da equação clássica de Boltzmann satisfaz as equações de conservação dos fluidos, sendo possível obter a ENS a partir daquela (BARDOS; UKAI, 1991).

A equação clássica de Boltzmann descreve uma função de distribuição em termos de interação microdinâmica. O desenvolvimento da equação baseia-se no fato de que um sistema pode ser descrito estatisticamente através de uma função de distribuição $f(x, e, t)$, onde $x$ e $e$ representam, respectivamente, os vetores de posição espacial e de velocidade das partículas em um instante de tempo $t$. Para um determinado instante $t$ o número de moléculas existentes em um espaço $dx$ com velocidade $de$ é definido por $f(x, e, t)\, dx\, de$. Considerando que não existem colisões entre as partículas, então em um instante de tempo $t + dt$ cada uma das partículas terá se movido de $x$ para $x + edt$ e a sua velocidade terá mudado de $e$ para $e + adt$, sendo $a$ a aceleração das partículas devido à ação de uma força externa. Como o número de moléculas se mantém constante, então:

$$f(x + edt, e + adt, t + dt)\, \Delta x\, \Delta e - f(x, e, t)\, \Delta x\, \Delta e = 0 \tag{2.6}$$

Entretanto, caso ocorram colisões entre as partículas, o número de partículas colididas será definido como a diferença entre $f(x + edt, e + adt, t + dt)\, \Delta x\, \Delta e$ e $f(x, e, t)\, \Delta x\, \Delta e$. Essa diferença pode ser definida como $\Omega(f(x, e, t))\, dx\, de\, dt$, onde $\Omega(f(x, e, t))$ é um operador de colisão. Dessa forma,

$$f(x + edt, e + adt, t + dt)\, \Delta x\, \Delta e - f(x, e, t)\, \Delta x\, \Delta e = \Omega(f(x, e, t))\, \Delta x\, \Delta e\, \Delta t \tag{2.7}$$

Dividindo-se a equação por $\Delta t\, \Delta x\, \Delta e$ e considerando que o limite de $\Delta t \to 0$, obtém-se a Equação Clássica de Boltzmann (BUICK, 1997):

$$\left( \frac{\partial}{\partial x} e + \frac{\partial}{\partial e} a + \frac{\partial}{\partial t} \right) f(x, e, t) = \Omega(f(x, e, t)) \tag{2.8}$$

A densidade $\rho$, a velocidade $v$ e a energia interna $\epsilon$ são obtidas, respectivamente, através da função de distribuição $f(x, e, t)$, sendo $m$ a massa molecular, conforme expressam as equações:

$$\rho(x, t) = \int m f(x, e, t) de \tag{2.9}$$

$$\rho(x, t) v(x, t) = \int m e f(x, e, t) de \tag{2.10}$$

$$\rho(x, t) \epsilon(x, t) = \frac{1}{2} \int m (e - u)^2 f(x, e, t) de \tag{2.11}$$

Para a solução da equação de Boltzmann é preciso encontrar um operador de colisão adequado e que obedeça às regras de conservação de massa, momento e energia, definidas

nas Equações 2.9, 2.10 e 2.11. Um operador de colisão simples pode ser obtido retendo apenas as propriedades qualitativas e a média das propriedades do operador de colisão real. A idéia baseia-se em um tempo de relaxação constante, conforme expressa o modelo BGK (Bhatnagar, Gross, Krook) (FLEKKøY, 1993):

$$\Omega = -\frac{f(x,e,t) - \overline{f}(x,e,t)}{\tau} \tag{2.12}$$

Nessa equação, $\tau$ é o tempo de relaxação (grandeza de tempo entre duas colisões) e $\overline{f}(x,e,t)$ é a função distribuição de probabilidade de equilíbrio local de Maxwell-Boltzmann (Teorema H) definido normalmente por:

$$\overline{f} = \frac{\rho}{m}\left(\frac{m}{2\pi k_B T}\right)^{\frac{3}{2}} exp\left(\frac{-m(e-v)^2}{2k_B T}\right) \tag{2.13}$$

Sendo $k_B$ a constante de Boltzmann, com o valor aproximado de $1,38065 \times 10^{-23}$ J/K, onde $\int_{-\infty}^{+\infty} \overline{f}(v)dv = 1$.

Uma vez definido o contexto da área, as equações macroscópicas que descrevem fluxos de fluidos e a base para o desenvolvimento do MLB, o próximo passo é apresentar o funcionamento do método. Para tanto, o Capítulo 3 expõe, de forma aprofundada, as equações, o modelo e o algoritmo do MLB proposto.

# 3 MÉTODO DE LATTICE BOLTZMANN

O Método de Lattice Boltzmann (MLB) tem sido freqüentemente utilizado como um modelo alternativo para a simulação computacional da Dinâmica de Fluidos definidas pelas equações de Navier-Stokes (SUCCI, 2001). Desenvolvido a partir da equação de transporte de Boltzmann, base da teoria cinética dos fluidos, ou através do método de LGA, o MLB é empregado em uma ampla variedade de pesquisas devido ao seu grande potencial para simulação de fluxos turbulentos, fluxos em múltiplas fases e fluxos com condições de contorno irregulares.

Este Capítulo descreve o MLB, apresentando as idéias de modelos para a representação de fluidos, as equações relacionadas e um algoritmo para o método. São ressaltadas ainda as diferenças do método em relação a outras abordagens, especialmente em relação a métodos de resolução diretos das equações de Navier-Stokes, como também o desenvolvimento do método em seus diversos aspectos. Para tanto, são discutidas inicialmente as características de Lattice Gas Automata e as modificações impostas a essa técnica que deram origem ao MLB.

## 3.1 Lattice Gas Automata

As ENS apresentadas no capítulo anterior, e que estão relacionadas com a abordagem microscópica através da EB, podem ser simuladas numericamente através do método de Lattice Gas Automata (LGA) (FRISCH et al., 1987). Esse método de simulação é obtido através da simplificação e da discretização da microdinâmica de um fluido, conforme ilustra a Figura 3.1. Desta forma, a distribuição das partículas passa a ocorrer sob um modelo regular e restrito, o que simplifica a simulação das partículas. As principais diferenças entre o modelo real e o LGA estão sumarizadas na Tabela 3.1.

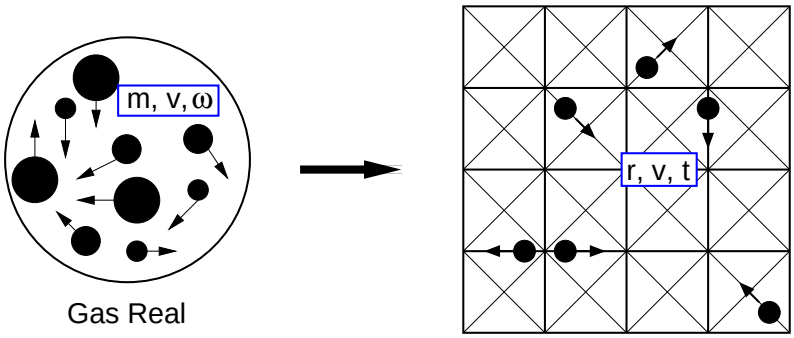


Figura 3.1: Passagem de um modelo microdinâmico para um modelo computacional

Tabela 3.1: Diferenças entre o modelo real e LGA

| **Característica** | Modelo Real | LGA |
|---|---|---|
| Massa das partículas | Diferentes | Iguais |
| Deslocamento das partículas | Aleatório e interativo | Iterações de propagação e de colisões simples |
| Momento | Contínuo | Discreto |

LGA pode ser entendido como uma classe de autômato celular aplicado a microdinâmica de fluidos, sendo, por isso, conhecido também como ***Lattice Gas Cellular Automata***. Um autômato celular é um modelo discreto que consiste de um conjunto infinito e regular de células distribuídas em uma malha, no qual cada uma das células possui um número finito de estados, sendo a malha formada por um número qualquer de dimensões finitas, conforme ilustra a Figura 3.2. As regras de propagação das células são definidas em função do estado das células vizinhas, sendo as mesmas válidas para todas as células. A atualização das células ocorre simultaneamente em intervalos de tempo discretos.

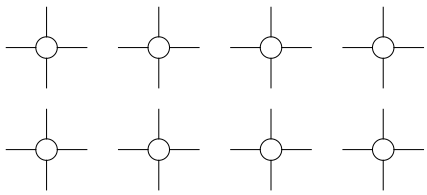

Figura 3.2: Esquema de um autômato celular

No caso dos fluidos, cada um dos estados das células representa uma característica física, como a direção da velocidade, que vai sendo alterada, conforme as definições do autômato. O uso de um autômato celular para a modelagem da estrutura microscópica de um fluido tem a vantagem de poder ser facilmente implementado computacionalmente, uma vez que o computador é uma máquina de estados e, portanto, capaz de processar de uma maneira simples o fluxo (troca de estados) de um fluido. Todas as células obedecem as mesmas regras, sendo atualizadas simultaneamente em tempos discretos. Em LGA e no MLB, os autômatos celulares utilizados para descrever um fluido são representados por estruturas conhecidas por reticulado.

## 3.2 Modelos de Reticulado

Na literatura existem diversos modelos de reticulado (QIAN; D'HUMIÈRES; LALLEMAND, 1992; WOLF-GLADROW, 2000). O primeiro modelo proposto é um reticulado quadrado bidimensional conhecido como HPP (*Hardy, Pazzis e Pomeau*) (HARDY; POMEAU; PAZZIS, 1973). Um reticulado quadrado possui somente quatro direções de deslocamento possíveis para cada partícula, conforme pode ser visto na Figura 3.3. Tanto a velocidade como a massa das partículas são definidos como valores unitários. Caso haja um choque entre as partículas, estas devem imediatamente ocupar os dois outros nós não ocupados anteriormente, conforme ilustra a Figura 3.4.

Uma deficiência desse modelo é de que a estrutura dos gases não é isotrópica, já que o modelo não garante invariância rotacional, o que impossibilita a obtenção de resultados reais precisos (WEI et al., 2004).

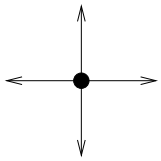

Figura 3.3: Modelo HPP

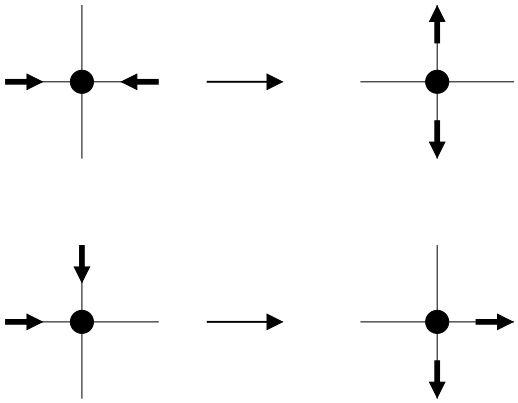

Figura 3.4: Regras de colisão do modelo HPP

Outro modelo conhecido como FHP (*Frisch, Hasslacher e Pomeau*), mas que garante a isotropia, baseia-se em um reticulado hexagonal bidimensional, conforme ilustra a Figura 3.5. Nele, cada partícula pode percorrer seis direções e, conseqüentemente, ter seis direções de velocidade diferentes (FRISCH et al., 1987). As diferentes regras de colisão para o modelo FHP são apresentadas na Figura 3.6.

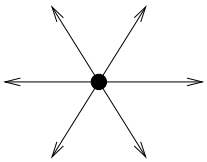

Figura 3.5: Modelo FHP

Para cada colisão de partículas em sentidos opostos, ocorre um desvio de $60^o$, mantendo, desta forma, o momento no estado seguinte à colisão igual a zero. Quando múltiplas possibilidades existirem, é feita uma escolha aleatória para a definição do novo estado.

Um modelo de reticulado geralmente recebe uma nomenclatura de acordo com a sua estrutura geométrica e o seu modelo de propagação de partículas. Assim, o modelo HPP visto anteriormente é conhecido também por D2Q5, uma vez que o mesmo é representado bidimensionalmente e possui 4 direções de deslocamento além de uma posição estática. Já o modelo FHP pode ser denominado de D2Q7, considerando o fato do mesmo ser bidimensional e com 6 direções de deslocamento mais uma posição estática.

Outros modelos bidimensionais podem apresentar um número maior de direções de deslocamento. A Figura 3.7 apresenta um modelo com 8 direções possíveis, sendo que as velocidades nas direções horizontais $(\pm 1, 0)$ e verticais $(0, \pm 1)$ são unitárias e a velocidade das diagonais $(\pm 1, \pm 1)$ são de $\sqrt{2}$.

Modelos mais avançados de reticulados podem ser obtidos em estruturas tridimensi-

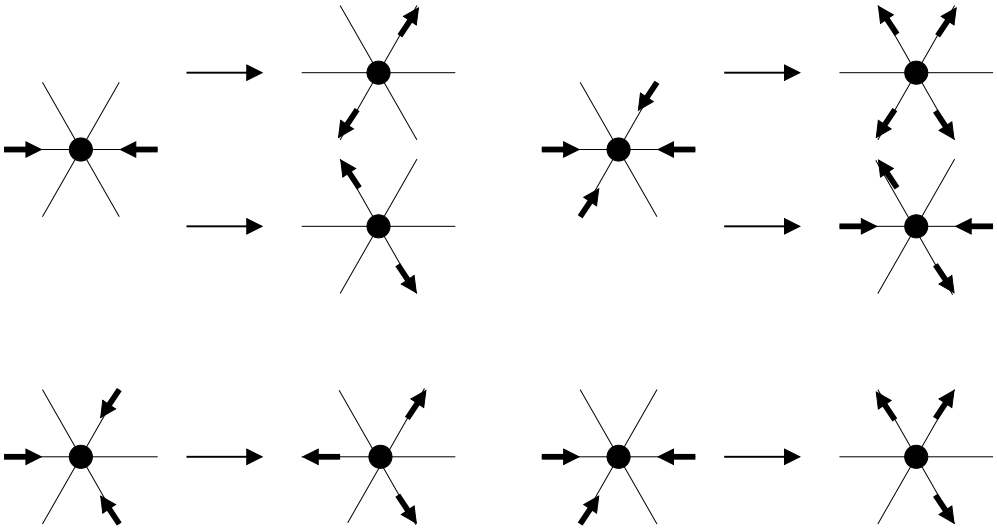

Figura 3.6: Regras de colisão do modelo FHP

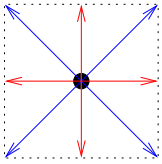

Figura 3.7: Modelo D2Q9

onais, com a definição de diferentes regras de colisão ou de velocidades de propagação para cada uma das direções. Cada um dos modelos apresenta as mesmas características básicas de funcionamento, conforme visto anteriormente nos modelos bidimensionais. Em um modelo tridimensional também é necessário que a geometria seja simétrica para satisfazer a isotropia requerida. Uma idéia simples para se alcançar isso é considerar que existem quatro tipo de sub-reticulados, os quais possibilitam um conjunto de 26 direções vizinhas. As Figuras 3.8, 3.9, 3.10 e 3.11 ilustram cada um dos sub-reticulados possíveis, recebendo a nomenclatura de $q_0$, $q_1$, $q_2$ e $q_3$.

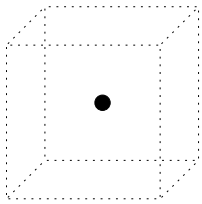

Figura 3.8: Sub-reticulado $q_0$

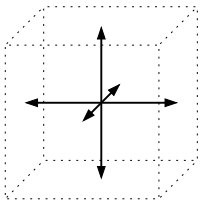

Figura 3.9: Sub-reticulado $q_1$

A combinação de cada um deles pode gerar diferentes modelos que são identificados através do número de dimensões (2D, 3D) e pelo número de possibilidades de deslocamento possíveis (Q15, Q19, Q27). Por exemplo, as combinações do sub-reticulados $q_0$, $q_1$, $q_3$ originam o modelo D3Q15. Já a combinação de todos eles gera o modelo D3Q27. Um dos modelos mais utilizados é o D3Q19, resultado da combinação de sub-reticulados $q_0$, $q_1$ e $q_3$. Cada um dos sub-reticulados também possui velocidades diferentes, conforme apresentado na Tabela 3.2.

Maiores informações referentes aos modelos de reticulado estão descritos na Tabela 3.3.

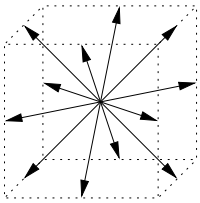

Figura 3.10: Sub-reticulado $q_2$

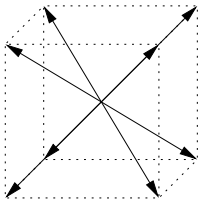

Figura 3.11: Sub-reticulado $q_3$

Tabela 3.2: Velocidade relativa das partículas para cada uma das direções do reticulado

| **Sub-reticulado** | **Velocidade** |
|---|---|
| $q_0$ | $0$ |
| $q_1$ | $1$ |
| $q_2$ | $\sqrt{2}$ |
| $q_3$ | $\sqrt{3}$ |

Nessa tabela estão relacionados os valores para a velocidade do som, energia e peso das arestas (quanto cada aresta interfere na densidade do ponto), respectivamente, para diferentes alguns modelos de reticulado uni- bi- e tridimensionais.

Tabela 3.3: Valores de algumas propriedades para diversos modelos de reticulado

| **Geometria** | $c_s^2$ | **Energia** | **Peso** |
|---|---|---|---|
| D1Q3 | 1/3 | 0 | 4/6 |
| | | 1/2 | 1/6 |
| D1Q5 | 1 | 0 | 6/12 |
| | | 1/2 | 2/12 |
| | | 2 | 1/12 |
| D2Q9 | 1/3 | 0 | 16/36 |
| | | 1/2 | 4/36 |
| | | 2 | 1/36 |
| D3Q15 | 1/3 | 0 | 16/72 |
| | | 1/2 | 8/72 |
| | | 2 | 1/72 |
| D3Q19 | 1/3 | 0 | 12/36 |
| | | 1/2 | 2/36 |
| | | 2 | 1/36 |

## 3.3 Equações de Lattice Gas Automata e Algoritmo

LGA apresenta-se como uma dinâmica molecular em que espaço, tempo e velocidade das partículas são discretos. Um estado $t$ de uma célula representando uma partícula consiste de regras que levam em consideração o estado $t-1$ da mesma, bem como o conjunto de células vizinhas. A representação de uma partícula é feita através de um conjunto de variáveis binárias $n_i(x,t)(i=1,...,d)$, em que cada $n_i(x,t)$ representa uma das $d$ direções de deslocamento possíveis (posições vizinhas) no reticulado. Cada $n_i(x,t)$

assume o valor unitário 1, caso exista deslocamento para a direção a qual ele representa, ou 0, caso não exista deslocamento.

A evolução da expressão é dada por:

$$n_i(x + e_i, t + 1) = n_i(x, t) + \Omega_i(n(x, t)), (i = 0, 1, ..., d) \tag{3.1}$$

Onde $e_i$ representa cada um dos deslocamentos possíveis de uma partícula. Cada $e_i$ possui um valor unitário associado à direção de deslocamento e um valor nulo para os demais casos onde não há deslocamento. Já o operador $\Omega$ define as regras de colisão das partículas.

Em cada passo do método ocorrem dois tipos de operações:

- A operação de propagação (deslocamento), onde as partículas deslocam-se para uma nova posição, conforme a direção definida por $e$. As partículas movem-se através das arestas do reticulado, sendo que somente uma partícula pode estar em um determinado lugar em um determinado instante.

- A operação de colisão, onde são estabelecidos os critérios de deslocamento, caso mais de uma partícula atinja um mesmo ponto. O operador de colisão define as mudanças que precisam ocorrer nos vetores de velocidade das partículas, de acordo com regras de dispersão pré-definidas.

Na Figura 3.12 são ilustradas as operações de deslocamento e colisão em um reticulado. Para tanto foi adotado um reticulado quadrado (quatro direções de deslocamento), mostrando o evolução das partículas de um fluido para três instantes de tempo consecutivos. No instante de tempo $t$ ocorre uma colisão entre duas partículas no centro do reticulado. Uma vez que duas partículas não podem ocupar o mesmo lugar no espaço, a colisão é tratada através da disposição das partículas em posições subseqüentes ao ponto de colisão, os quais se encontram livres.

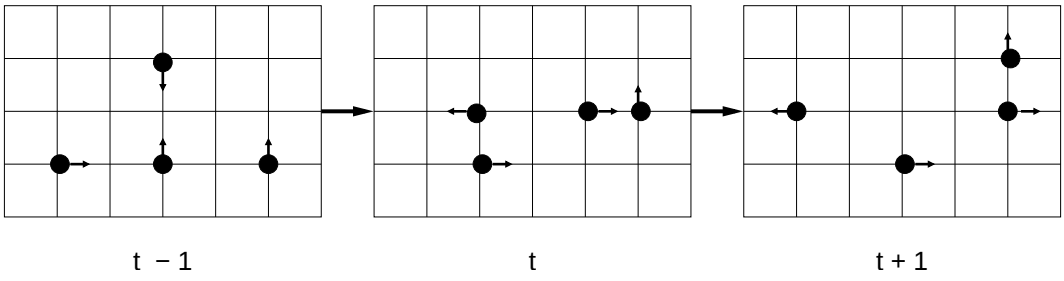


Figura 3.12: Ilustração do modelo de propagação e colisão em LGA

A partir da distribuição de partículas é possível derivar valores macroscópicos para algumas propriedades físicas. Por exemplo, o valor da densidade $\rho$ pode ser obtido através da soma dos elementos ($elem$) pelo número de sítios (regiões onde os elementos podem estar) existentes:

$$\rho = \sum elem / \sum sitios \tag{3.2}$$

E as velocidades médias para um determinado eixo de direção coordenado ($desl\_x$ e $desl\_y$):

$$v_x = \sum desl_x / \rho \tag{3.3}$$

$$v_y = \sum desl_y / \rho \tag{3.4}$$

No caso da Figura 3.12 para o momento $n$, o valor de $\rho$ é definido como:

$$\rho = 4/35 \tag{3.5}$$

Considerando positivas as velocidades que partem da esquerda para a direita para o eixo x e de baixo para cima para o eixo y, as velocidades $v_x$ e $v_y$ seriam, respectivamente:

$$v_x = (1 + 1 - 1)/\rho = 35/4 \tag{3.6}$$

$$v_y = (1)/\rho = 35/4 \tag{3.7}$$

Neste exemplo simples, duas direções são usadas. Nos casos reais discutidos na seção 3.2, muito mais direções de deslocamento podem ser usados.

As condições de contorno são definidas através de pontos especiais no reticulado através da técnica de ***Marker and Cell***. Cada um desses pontos formam os obstáculos pelos quais é delimitado a estrutura do fluxo. Diferentes regras de colisão podem ser aplicadas para esses pontos através de modificações no operador de colisão. Tais regras irão depender do tipo de fluido e das condições de contorno tratadas.

O algoritmo para LGA pode ser definido como um laço iterativo em busca de um estado de equilíbrio para uma determinada distribuição de partículas. Em cada fase é feito o processo de propagação e colisão, conforme mostrado anteriormente na Figura 3.12, mais o tratamento das condições de contorno. A partir da distribuição das partículas são calculadas as propriedades macroscópicas que se tem interesse.

LGA apresenta vantagens e desvantagens de uso e implementação. A Tabela 3.4 apresenta um quadro destacando os principais pontos para cada caso. Um dos aspectos negativos é a geração de erros estatísticos no cálculo das propriedades físicas analisadas. Isto se deve ao fato de que as partículas realizam apenas operações binárias. Uma maneira de suavizar essas falhas é considerar uma média dos valores de espaço e tempo. Assim, por exemplo, para um determinado fluxo em um espaço bidimensional poderia ser utilizado um número maior de células vizinhas (64, 128, 256, 512, ...). No entanto, isso limita o espaço de resolução que pode ser obtido, além de haver a necessidade de uma quantia maior de informações para o cálculo dos valores de um determinado ponto.

Tabela 3.4: Vantagens e desvantagens de LGA

| **Vantagem** | **Desvantagem** |
|---|---|
| Integração simples e eficiente para qualquer geometria complexa | Erros estatísticos, longos tempos para o cálculo da média |
| Método local, fácil paralelização | Efeitos para tamanhos finitos, necessidade de reticulados grandes |
| Algoritmo eficiente: representação de partículas por *bits* ou inteiros | Dificuldade de controlar os parâmetros de fluxo |

## 3.4 Descrição Informal do Método de Lattice Boltzmann

Historicamente o MLB surgiu a partir de LGA. A principal diferença do MLB em relação ao modelo de LGA é substituir as variáveis *booleanas*, utilizadas para representar

a distribuição das partículas, por funções de distribuição reais para todas as partículas individuais, e ignorar o movimento individual das partículas e as correlações entre elas nas equações cinéticas (CHEN; DOOLEN, 1998). Esse procedimento elimina, assim, o uso da parte estatística. Por outro lado, os modelos de representação para os reticulados são os mesmos em ambos os métodos.

O MLB pode ser definido com um método numérico iterativo para a modelagem e simulação de propriedades físicas em fluidos (SUCCI, 2001; CHEN; DOOLEN, 1998; WOLF-GLADROW, 2000). Diferente de outros modelos numéricos, baseados na discretização de equações macroscópicas contínuas, tais como o método dos Elementos Finitos ou o método dos Volumes Finitos, o MLB está fundamentado em modelos microscópicos e equações de velocidade mesoscópica.

A principal idéia do MLB é construir um modelo cinético simplificado, que incorpore a essência física dos processos microscópicos ou mesoscópicos, a fim de que a média dessas propriedades obedeça à equação macroscópica desejada. A dinâmica macroscópica de um fluido é o resultado da coleção de estruturas microscópicas de um sistema. No entanto, ela não é sensível aos detalhes existentes na dinâmica microscópica. Esse é um dos motivos para o uso de métodos simplificados em lugar de fluxos macroscópicos. Através do desenvolvimento de uma versão simplificada da equação cinética, busca-se evitar a resolução de equações complicadas, como a equação completa de Boltzmann, ou considerar cada partícula como uma simulação molecular.

Embora o MLB seja baseado em um representação de partículas, seu principal foco é a obtenção da média da estrutura macroscópica. A equação cinética do método prove muitas das vantagens da dinâmica molecular, incluindo uma clara representação física, simplicidade de implementação das condições de contorno e algoritmos completamente paralelos, já que as operações são estritamente locais.

Segundo Chen (CHEN; DOOLEN, 1998), a natureza cinética do MLB introduz três importantes características que o distinguem de outros métodos numéricos. O primeiro ponto consiste no fato de que as operações de convecção (deslocamento das partículas) são baseadas em operações lineares, o que diverge do proposto por modelos macroscópicos. Uma segundo ponto diz respeito à conservação do momento linear, que pode ser obtida pelo MLB através de uma equação de estados, sendo esse valor igual ao obtido pela resolução das ENS. O terceiro ponto trata das direções de velocidade do MLB, que são definidas por um conjunto limitado de direções, diferente do que ocorre na EB.

## 3.5 Equação de Lattice Boltzmann

A Equação de Lattice Boltzmann (ELB) pode ser obtida de diferentes maneiras. Ela pode ser vista tanto como um modelo de velocidade discreto a partir da evolução de LGA, como também uma derivação do modelo físico através da Equação Cinética de Boltzmann (WOLF-GLADROW, 2000). Na seqüência são apresentados alguns passos para a obtenção da ELB a partir da EB.

Negligenciando forças externas, a função de propagação $f(x, e, t)$ pode ser expressa pela Equação de Boltzmann como:

$$\frac{\partial f}{\partial t} + e\frac{\partial f}{\partial x} = Q(f) \tag{3.8}$$

O termo de relaxação $Q(f)$ é quadrático em $f$, consistindo originalmente de uma expressão integral-diferencial complexa. Uma simplificação apropriada da integral de colisão para o estado de quase-equilíbrio em hidrodinâmicas com baixo número de *Mach* é a

aproximação via tempo de relaxação simples, conhecida como modelo BGK (Bhatnagar-Gross-Krook) (BHATNAGAR; GROSS; KROOK, 1954):

$$Q(f) = -\frac{1}{\lambda}\left(f - \overline{f}\right) \tag{3.9}$$

Onde $\overline{f}$ é a função de distribuição de equilíbrio de Maxwell-Boltzmann e $\lambda$ é um tempo de relaxação que controla a taxa de aproximação do equilíbrio, ou seja, a viscosidade do fluido. A relaxação BGK cumpre ainda o teorema H de Boltzmann e a conservação local de massa e momento.

Para determinar $f$ numericamente, primeiramente ela é discretizada em relação ao campo da velocidade, utilizando um conjunto finito de vetores de velocidade $e_i (i = 0, ..., d)$, gerando a equação de velocidade discreta de Boltzmann:

$$\frac{\partial f_i}{\partial t} + e_i \frac{\partial f_i}{\partial x} = -\frac{1}{\lambda}(f_i - \overline{f_i}), i = 0, ..., d\,, \tag{3.10}$$

onde $f_i(x,t)$ é equivalente a $f(x, e_i, t)$ e $\overline{f_i}$ é a função de distribuição de equilíbrio discreto.

Os modelos de reticulado usados em LGA vistos na Seção 3.2 são igualmente utilizados no MLB. Para a simulação de fluxos bidimensionais o principal modelo de simulação utilizado é o D2Q9. Já para a simulação tridimensional, tanto o modelo D3Q15, como o modelo D3Q19 são amplamente utilizados. Apesar da discretização definir um pequeno número de direções de deslocamento para esses modelos, elas são suficientes para descrever um fluido próximo do estado de equilíbrio para hidrodinâmicas com baixos valores de *Mach*. Para todos esses modelos a função $\overline{f_i}$ apropriada para a distribuição de equilíbrio possui a forma:

$$\overline{f_i} = \rho w_i \left[1 + \frac{3}{c^2} e_i.u + \frac{9}{2c^4}(e_i.u)^2 - \frac{3}{2c^2} u.u\right] \tag{3.11}$$

Com $c = \Delta x/\Delta t$ e vetores de velocidade de partícula discretos $e_i$. O peso do fator $w_i$ depende somente do modelo do reticulado e é apresentado na Tabela 3.5 para os três modelos 3D mais utilizados. A função de distribuição de equilíbrio discreta $\overline{f_i}$ é derivada da função de distribuição $\overline{f}$ de equilíbrio de Maxwell-Boltzmann de tal maneira que os momentos de velocidade até a quarta ordem são idênticos à $\overline{f}$.

Tabela 3.5: Peso dos fatores $w_i$ para os vetores de velocidades discretas $e_i$ de diferentes modelos de reticulado

| **Modelo** | $\|e_i\|^2 = 0$ | $\|e_i\|^2 = 1$ | $\|e_i\|^2 = 2$ | $\|e_i\|^2 = 3$ |
|---|---|---|---|---|
| D2Q9 | $w_i$ = 4/9 | $w_i$ = 1/9 | $w_i$ = 1/36 | |
| D3Q15 | $w_i$ = 2/9 | $w_i$ = 1/9 | | $w_i$ = 1/72 |
| D3Q19 | $w_i$ = 1/3 | $w_i$ = 1/18 | $w_i$ = 1/36 | |

Os valores macroscópicos da densidade $\rho$, momento $\rho u$ e de tensor de fluxo de momento $\Pi_{\alpha\beta}$, sendo $N$ o número de partículas e $\sum_j = \sum_i^d$, podem ser desenvolvidos como:

$$\rho = \sum_{j=0}^{N} f_j \simeq \sum_{j=0}^{N} \overline{f_j} \tag{3.12}$$

$$\rho v = \sum_{j=0}^{N} e_j f_j \simeq \sum_{j=0}^{N} e_j \overline{f_j} \tag{3.13}$$

$$\Pi_{\alpha\beta} = \sum_{j=0}^{N} e_{j\alpha} e_{j\beta} f_j \tag{3.14}$$

Já a velocidade do som $c_s$ desses modelos é de:

$$c_s = c/\sqrt{3} \tag{3.15}$$

E a pressão $p$ é definida pela equação de estado de um gás ideal:

$$p = \rho c_s^2 \tag{3.16}$$

A discretização do espaço e do tempo é feita apropriadamente através de uma aproximação de diferenças finitas. A forma explicita das equações discretizadas é dada por:

$$f_i(x + e_i \Delta t, t + \Delta t) - f_i(x,t)) = -\frac{1}{\tau}(f_i(x,t) - \overline{f_i}(x,t)) \tag{3.17}$$

Onde $\tau = \lambda/\Delta t$ é o tempo de relaxação adimensional. Assim, o MLB pode ser considerado também como uma forma especial de aplicação da técnica das diferenças finitas para a equação cinética, onde a função de distribuição da velocidade é discreta (WOLF-GLADROW, 2000).

## 3.6 Condições de Contorno

As condições de contorno do MLB não possuem relação com as condições de contorno macroscópicas uma vez que no método as mesmas são cinéticas. No entanto, definindo-se apropriadamente regras microscópicas é possível alcançar a estrutura macroscópica. Uma das soluções mais simples para a simulação de bordas rígidas é o uso do mecanismo conhecido como ***Bounce Back*** (barreiras reflexivas). A idéia consiste em inverter a direção de velocidade para as partículas das bordas de maneira que:

$$f_i^{out}(x,t) = f_i^{in}(x,t) \tag{3.18}$$

Assim, $e_i = -e_i$, ou seja, $e_i$ é rotacionado para cada direção de velocidade da borda, garantindo o retorno para o fluido na próxima iteração do laço, com o momento contrário. A Figura 3.13, ilustra o funcionamento do mecanismo de *Bounce Back*, aplicado a um determinado ponto em dois instantes de tempo consecutivos.

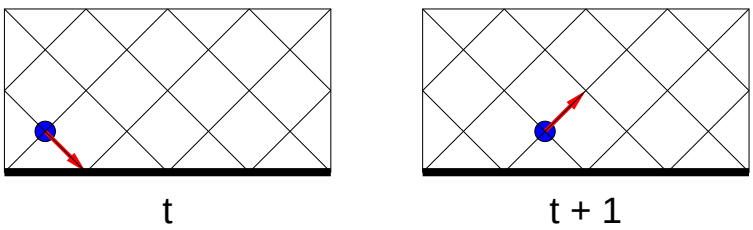


Figura 3.13: Exemplo de funcionamento do mecanismo de *Bounce Back*

Isso garante que o fluxo não ultrapassa as bordas, tendo velocidades nulas nessas regiões.

## 3.7 Relação entre a Equação de Lattice Boltzmann e a Equação Cinética de Navier-Stokes

A partir da ELB é possível determinar também o comportamento de um fluxo no limite macroscópico de acordo com a definição das ENS (HE; LUO, 1997b). As ENS podem ser derivadas formalmente a partir da ELB através de uma expansão multi-escala padrão, via expansão de Chapman-Enskog, com o tempo e o espaço reescalados baseados no número de Knudsen $\varepsilon$ fazendo, respectivamente:

$$t_1 = \varepsilon t,\ t_2 = \varepsilon^2 t,\ x_1 = \varepsilon x,\ \frac{\partial}{\partial t} = \varepsilon \frac{\partial}{\partial t_1} + \varepsilon^2 \frac{\partial}{\partial t_2},\ \frac{\partial}{\partial x} = \varepsilon \frac{\partial}{\partial x_1} \tag{3.19}$$

E a função de distribuição $f_i$ expandida como uma equação de segunda ordem de precisão (HE; LUO, 1997b; FRISCH et al., 1987):

$$f = f^{(0)} + \varepsilon f^{(1)} + \varepsilon^2 f^{(2)} + O(\varepsilon^3) \tag{3.20}$$

No limite de um fluxo incompressível, onde $|v|/c_s < 1$, a lei de conservação de massa e momento é de, respectivamente:

$$\frac{\partial v_i}{\partial x_i} = 0 \tag{3.21}$$

$$\frac{\partial v}{\partial t} + v\frac{\partial v}{\partial x} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \mu \nabla^2 v \tag{3.22}$$

Ainda usando o resultado da expansão de Chapman-Enskog, é possível obter a relação entre o tempo de relaxação $\tau$ e a viscosidade cinética $\mu$. A relação é definida por:

$$\mu = (\tau - 1/2) c_s^2 \Delta t \tag{3.23}$$

Como a viscosidade é positiva, $\mu$ precisa ser maior que $1/2$ em todos os casos. A equação 3.23 difere de $\mu = \lambda c_s^2$ para corrigir o erro de truncamento da discretização da ELB.

A ELB difere de métodos diretamente baseados das ENS em diversos aspectos. Em abordagens tradicionais de DFC, a modelagem física da estrutura de um fluido é a aproximação numérica são dois procedimentos separados, enquanto no MLB, essas duas fases são combinadas em conjunto. A principal diferença entre MLB e abordagens numéricas para a resolução da ENS é que no MLB as regras de transição física são discretas, enquanto nas outras a discretização é feita no nível macroscópico da ENS. No entanto, em ambos os casos, as mesmas leis de conservação física são satisfeitas.

A seguir são descritos outros pontos, conforme apresentado em (YU et al., 2003), que tratam da diferença entre as duas técnicas:

- Enquanto a equação cinética de Navier-Sokes é uma EDP de segunda ordem, no MLB a discretização da EB resulta em um conjunto de EDPs. Isso reflete no uso de diferentes formas de discretização para cada tipo de equação.

- Métodos de resolução das ENS inevitavelmente precisam tratar dos termos convectivos não lineares $v.\nabla v$. O MLB evita totalmente a resolução desse termo porque a convecção torna-se uma operação vetorial (linear). O termo não linear das ENS é ocultado nos termos quadráticos da velocidade da função de distribuição do equilíbrio.

- Métodos para a resolução da equação cinética incompressível de Navier-Stokes precisam resolver a equação de Poisson para o cálculo da pressão. No MLB a pressão é obtida através de uma equação de estado.

- Condições de contorno em geometrias complexas exigem um tratamento cuidadoso em ambos modelos de simulação. Em métodos de resolução baseados na resolução discreta das ENS os componentes de pressão normal e paralela requerem a manipulação apropriada de estimativas geométricas de normais e tangentes e de extrapolações lineares. No caso do MLB o problema das condições de contorno surge porque o modelo contínuo não apresenta um equivalente direto.

- O MLB explora melhor problemas em escala microscópica, uma vez que a incorporação física das iterações em nível molecular da EB é mais simples.

- A discretização do MLB é feita pela discretização do domínio de velocidade da partícula. A velocidade discretizada e a configuração do espaço leva a uma malha regular quadrada. Isso é uma limitação para determinadas aplicações, mesmo com o uso de técnicas de refinamento, o que exige uma implementação especial para as bordas.

- O MLB não é eficiente para a solução de problemas de estados estáveis porque a velocidade convergência da propagação é bastante lenta, uma vez que as unidades de variação de tempo são unitárias.

Para efeitos de comparação, as principais diferenças apontadas anteriormente entre as ENS e o MLB são apresentadas na Tabela 3.6.

Tabela 3.6: Principais diferenças entre a resolução utilizando as ENS e o MLB

| **Característica** | **ENS** | **ELB** |
|---|---|---|
| Equação de velocidade | EDP de segunda ordem | Conjunto de EDPs de primeira ordem |
| Termo convectivo não linear $v.\nabla v$ | É tratado | Substituído por uma operação vetorial (linear) |
| Cálculo da pressão | Resolução da Equação de Poisson | Equação de Estado |
| Condições de contorno em Geometrias complexas | Estimativas geométricas e extrapolação | Não apresenta um equivalente direto para o modelo contínuo |
| Escala Microscópica | Não é o objetivo do modelo | Representação mais simples |
| Discretização | Das equações | Da velocidade das partículas |
| Problemas com estados estáveis | Indiferente | Necessidade de muitas iterações |

## 3.8 Algoritmo

O algoritmo do MLB possui uma estrutura similar ao método de Jacobi, o qual é utilizado para a solução iterativa de sistemas lineares em malhas estruturadas (BARRETT

et al., 1994; KÖRNER et al., 2005). Nesse caso, pode-se dizer que o laço de repetição do método de Jacobi coincide com o laço iterativo controlado pelo tempo discreto do MLB. O algoritmo para o MLB pode ser descrito como:

1. Determinar as condições iniciais para todos os pontos da grade, escolher adequadamente a densidade $\rho$ e a velocidade $e$ das células externas (condições de contorno do fluido) e definir a escala do tempo de relaxação $\tau$.

2. Calcular a densidade $\rho$ e velocidade $u$ das variáveis macroscópicas para cada célula usando as Equações 3.12 e 3.13.

3. Calcular os valores da função de equilíbrio usando a Equação 3.11 e, a partir disso, obter os valores da função de relaxação através da Equação 3.9;

4. Utilizar o valor de equilíbrio para calcular a função de distribuição para cada ponto, segundo a Equação 3.17;

5. Propagar a distribuição das partículas para todas as células vizinhas;

6. Modificar a distribuição local dos pontos para satisfazer as condições de contorno através da Equação 3.18;

7. Retornar ao passo 2 enquanto o tempo de execução ou o número de iterações for menor que o máximo estimado.

As principais operações do laço de repetição do MLB estão ilustradas na Figura 3.14.

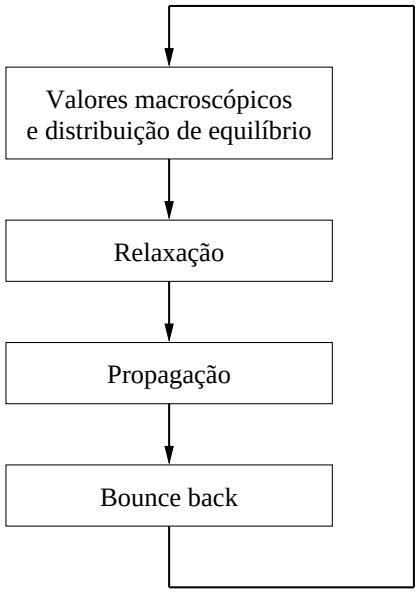


Figura 3.14: Estrutura do algoritmo do MLB

## 3.9 Áreas de Aplicação

As características mencionadas anteriormente mostram que o MLB é indicado para uma série de aplicações com diversos graus de dificuldade. Na literatura existem diversas

implementações de problemas com condições de contorno simples (CHEN; DOOLEN, 1998). Exemplos desse tipo de problemas são encontrados na passagem de fluidos por aberturas ou na análise do comportamento de fluidos perante a presença de obstáculos (objeto sólidos).

Uma característica especial do método é de que condições de contorno irregulares podem ser representadas de uma maneira simples, além de consumir pouco tempo computacional para o cálculo das bordas. Por causa disso, o MLB é indicado para a simulação de fluidos em geometrias complicadas, como ocorre em superfícies porosas, onde é mais difícil descrever a interação entre o fluido e o sólido que o contém. Outro fato, é de que o MLB, diferente de LGA, pode ser usado para fluidos com baixa viscosidade. Desta forma, é possível realizar a simulação numérica direta de fluidos turbulentos e não homogêneos.

Outro ponto a ser considerado é a simulação de fluidos com múltiplas fases e multicomponentes. Devido à dificuldade em modelar o dinamismo e a importância que esse tipo de aplicações tem para as áreas da engenharia (superfícies porosas, dinâmica da ebulição, suspensão de partículas em fluidos e fluxos turbulentos), o MLB surge como uma alternativa, especialmente para modelos tridimensionais. Outras aplicações que fazem uso do MLB são encontrados na reação química de fluidos, em processos de combustão, na magnetohidrodinâmica (estudo das interações entre o eletromagnetismo e a hidrodinâmica), na transferência de calor em processos de reação-difusão e em processos de cristalização.

## 3.10 Paralelização do Método de Lattice Boltzmann

O MLB pode ser paralelizado através do particionamento do reticulado. Essa característica é muito importante, uma vez que a quantidade de memória e de processamento necessários para a simulação de ambientes reais é muito alta. Comparando o MLB com outras técnicas de simulação de fluidos, o custo de execução do método é relativamente maior (SUCCI, 2001). A existência de arquiteturas paralelas e de diferentes técnicas de programação, possibilitam alcançar excelentes níveis de desempenho para o método. Assim, no Capítulo 4 são aprofundadas as técnicas de paralelização e distribuição de dados empregados na paralelização do MLB, apresentando, ainda, os trabalhos relacionados.

# 4 ESTADO DA ARTE EM TÉCNICAS DE PARALELIZAÇÃO

O processo de desenvolvimento de aplicações tem sido simplificado pela existência de mecanismos de programação paralela padronizados e com uma vasta gama de recursos. Um dos recursos amplamente utilizado para tal finalidade é a biblioteca de comunicação *Message Passing Interface* (MPI) (MPI FORUM, 1994; GROPP et al., 1996). MPI possui um grande número de funções que podem ser utilizadas, tanto em implementações paralelas, como em implementações distribuídas. Entre os recursos encontram-se mecanismos de comunicação cartesiana, que possibilitam o endereçamento de mensagens aos processos segundo as posições atribuídas aos mesmos, definidas a partir de uma distribuição cartesiana. Tais recursos são imprescindíveis para a obtenção de uma boa eficiência paralela, tendo sido recorrente a sua utilização. Nesse contexto, este Capítulo tem por objetivo apresentar o estado da arte em técnicas de paralelização. Para tanto, é feito inicialmente uma rápida descrição da biblioteca MPI. Na seqüência são tratadas as técnicas de particionamento em blocos, como também os trabalhos relacionados à paralelização do MLB. Por fim, é feito um posicionamento, ressaltando o foco deste trabalho.

## 4.1 Message Passing Interface

*Message Passing Interface* (MPI) é um padrão para comunicação de dados na computação paralela (GROPP et al., 1996). Seu principal objetivo é disponibilizar uma interface que seja largamente utilizada no desenvolvimento de programas baseados em troca de mensagens. Além de garantir a portabilidade dos programas paralelos, a interface MPI possibilita a implementação eficiente de sua especificação para diversos tipos de máquinas paralelas existentes.

MPI define apenas o modelo de troca de mensagens tais como nomes de funções, seqüências de chamadas e resultados de subrotinas, sem se preocupar com a implementação em si. Por isso, existem diversas implementações do padrão MPI, cada qual com suas características e otimizações de código (LAM/MPI PARALLEL COMPUTING, 2006; OPEN MPI: Open Source High Performance Computing, 2006; MPICH HOME PAGE, 2006; MPICH2 HOME PAGE, 2006). Considerado uma evolução de *Parallel Virtual Machine* (PVM) (GEIST et al., 1994), a biblioteca pode ser utilizada em programas escritos em linguagem FORTRAN, C ou C++.

No padrão MPI, uma aplicação é constituída por um ou mais processos que podem ser executados em máquinas distintas, os quais se comunicam através de funções de envio e recebimento de mensagens via interface de rede. Assim, as implementações do padrão oferecem uma infraestrutura para a computação paralela na qual é possível trocar infor-

mações entre vários processadores. Tais recursos são ideais para modelos de programação *Single Program Multiple Data* (SPMD) e *Multiple Program Multiple Data* (MPMD), ou mesmo um modelo Mestre-Escravo.

MPI disponibiliza diferentes formas de comunicação. Os mecanismos de comunicação mais simples que podem ser utilizados são a comunicação ponto a ponto, onde ocorrem operações de troca de mensagens de um determinado processo com outro. Estruturas mais refinadas de comunicação são obtidas usando um grupo de processos que invocam operações coletivas (*collective*) de comunicação para a execução de operações globais. Além disso, MPI é capaz de suportar comunicação assíncrona e programação modular, através de mecanismos de comunicadores (*communicator*). Os comunicadores permitem ao usuário MPI definir módulos que encapsulem estruturas de comunicação interna (*group communications*).

O funcionamento básico de um programa com MPI consiste em lançar um conjunto de processos, os quais atuam sobre um determinado código fonte. Cabe ao programador diferenciar através de um identificador de processo quem é o responsável pela execução de cada parte do código. É através desse mesmo indicador que os demais processos podem acessar esse processo específico. Desta forma, é possível realizar a troca de mensagens, garantindo a comunicação entre os processo.

Os recursos encontrados em MPI são muito importantes pois garantem implementações paralelas com mecanismos de comunicação eficientes e uma maior independência entre as execuções dos processos. Além desses, MPI possui ainda mecanismos para a criação de estruturas cartesianas, bem como as funções necessárias para o mapeamento e acesso aos processos. Analisando-se esses recursos, além dos encontrados anteriormente, MPI oferece as condições ideais para a programação paralela baseada em particionamento de dados em blocos.

## 4.2 Particionamento em Blocos

Um dos grandes desafios da programação paralela é dividir da melhor forma possível um programa seqüencial em partes, a fim de que cada uma delas seja executada em paralelo. Para tanto, é necessário conhecer a estrutura do algoritmo, bem como a arquitetura na qual o programa paralelo irá ser executado. Dentre as estratégias de decomposição de programas adotadas para tal finalidade, a Decomposição de Tarefas e a Decomposição de Dados são freqüentemente utilizadas em arquiteturas com memória distribuída. No entanto, em termos de aplicações numéricas a Decomposição de Dados é a mais utilizada.

Em muitas aplicações paralelas a distribuição dos dados é feita de forma simples para todos os processos. Geralmente isso ocorre dividindo-se o conjunto de dados em fatias pelo número total de processadores existentes. Entretanto, essa estratégia não considera a dependência de dados existente no conjunto dos dados. O mesmo ocorre quando os dados são divididos em pequenas partes e, então, distribuídos ciclicamente por todos os processos. A Figura 4.1 ilustra a divisão de um conjunto de dados proporcionalmente por 6 processos e uma distribuição cíclica do mesmo conjunto de dados utilizando 4 processos.

Uma maneira eficiente para distribuir os dados entre os processadores paralelos é utilizar o particionamento em blocos (LAM; ROTHBERG; WOLF, 1991; JONSSON; KÅGSTRÖM, 2002; ELMROTH et al., 2004). O particionamento em blocos é definido como a divisão dos dados em mais de uma dimensão de estruturas multidimensionais de dados. Assim, um conjunto de dados que forma uma estrutura tridimensional, por

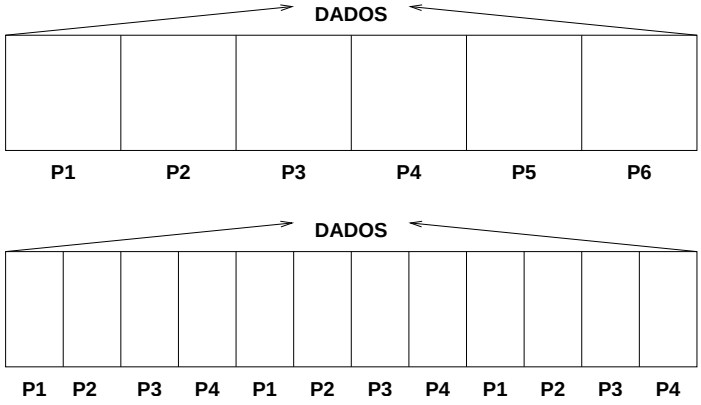


Figura 4.1: Exemplos de particionamento de dados: forma proporcional e forma cíclica

exemplo, pode ser dividido em blocos tanto em duas como em três dimensões. As Figuras 4.2 e 4.3 ilustram, respectivamente, a divisão dos dados em blocos bi- e tridimensionais. Através dessas figuras é possível perceber que a divisão em múltiplas dimensões pode ser obtida como combinação das estratégias de particionamento unidimensional.

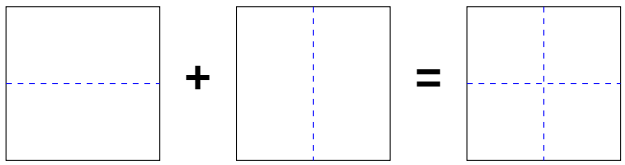


Figura 4.2: Exemplos de particionamento dos dados para o modelo bidimensional

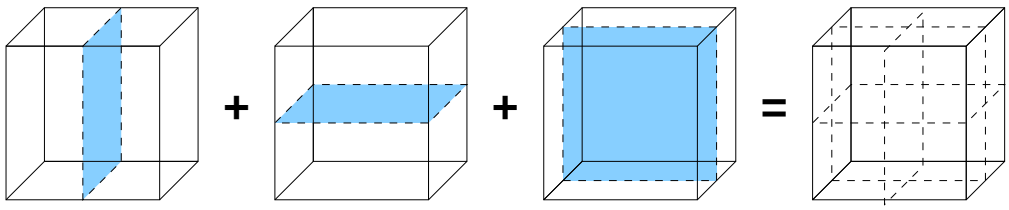


Figura 4.3: Exemplos de particionamento dos dados para o modelo tridimensional

O particionamento dos dados em bloco explora melhor os recursos computacionais disponíveis. Através dessa técnica é possível melhorar o acesso dos dados em memória e a taxa de acerto da memória *cache* por parte das aplicações (LAM; ROTHBERG; WOLF, 1991). O particionamento em blocos diminui também a dependência de dados que normalmente existe entre as regiões particionadas de forma unidimensional. Assim, os custos de comunicação e dependências de dado em sistemas com memória distribuída acabam sendo reduzidos.

Estratégias de particionamento de dados em blocos são freqüentemente adotadas em operações numéricas paralelas como forma de agregar desempenho as operações. Por causa disso, muitas propostas podem ser encontradas, especialmente para métodos de resolução de sistemas lineares (DONGARRA et al., 2002). As modificações feitas sobre técnicas de resolução tradicionais tornaram possível o surgimento de novos algoritmos, os quais são mais indicados para as arquiteturas de computadores atualmente existentes. Cabe destacar também que o algoritmo de resolução de sistemas lineares Linpack,

utiliza o particionamento em blocos para explorar ao máximo as arquiteturas paralelas, servindo, por isso, como *benchmark* de avaliação de desempenho de máquinas paralelas (DONGARRA, 1988).

## 4.3 Implementações Paralelas do Método de Lattice Boltzmann

O MLB tem sido utilizado em diversas situações como técnica de simulação de fenômenos físicos, especialmente da dinâmica de fluidos. Exemplos de trabalhos onde o método é aplicado são encontrados na modelagem de rochas-reservatório de petróleo (SANTOS et al., 2005; PHILIPPI; WOLF; SANTOS, 2005), sistema circulatório sanguíneo (FANG et al., 2002; KRAFCZYK et al., 1998), fenômenos gasosos (WEI et al., 2004), nanofluidos (XUAN; YU; LI, 2005), polímeros líquidos (ONISHI; CHEN; OHASHI, 2005), entre outros.

Uma das características freqüentemente ressaltadas na literatura do MLB é a possibilidade de paralelização de suas operações (SUCCI, 2001; KÖRNER et al., 2005). Boa parte dos autores considera essa abordagem uma das melhores formas de proporcionar desempenho às implementações. Assim, existem vários trabalhos relevantes que adotam um código paralelo para a elaboração de um programa eficiente, conforme pode ser observado na bibliografia relacionada sobre o assunto (PAN; PRINS; MILLER, 2004; SLOOT et al., 2004; DUPUIS, 2002; DESPLAT; PAGONABARRAGA; BLADON, 2001). Entretanto, a maior parte deles está voltado à solução eficiente de problemas específicos.

Uma exceção à utilização restrita do MLB em trabalhos acadêmicos é encontrada no *software* PowerFLOW (Exa Corporation, 2006). PowerFLOW é uma aplicação comercial para a simulação de vários tipos de estruturas físicas, especialmente aerodinâmicas (LOCKARD; LUO; SINGER, 2000). O *software* apresenta muitos recursos de simulação, incluindo fases de configuração, simulação propriamente dita e operações de pós-processamento. Em termos de simulação PowerFLOW está baseado em implementações de diferentes modelos do MLB. As implementações também estão disponíveis em versões paralelas, tendo sido paralelizadas com MPI. Por ser uma aplicação comercial, nenhum detalhe de implementação é apresentado, sendo apenas mencionado que o desempenho paralelo chega a atingir excelentes níveis.

A maior parte dos resultados publicados envolvendo o MLB estão geralmente concentrados na análise das soluções físicas obtidas pelas aplicações. No entanto, algumas exceções são encontradas (CARTER; OLIKER, 2006; POHL et al., 2004; KÖRNER et al., 2005).

Um exemplo de avaliação de desempenho paralelo do MLB encontra-se em (CARTER; OLIKER, 2006). Nesse artigo são apresentadas comparações de performance obtidas em algumas arquiteturas vetoriais (Cray X1, NEC SX-6 e NEC SX-8) e superescalares (IBM Power3, Intel Itanium2 e AMD Opteron) contemporâneas. O estudo de caso foi desenvolvido a partir de uma implementação paralela bidimensional e outra tridimensional do MLB, as quais são empregadas em um problema de turbulência da área da hidrodinâmica de magnetos. Os principais resultados apresentados demonstram um nível de desempenho paralelo muito superior das arquiteturas vetoriais em relação ao desempenho obtido para as arquiteturas escalares. As arquiteturas vetoriais tiveram uma maior escalabilidade, atingindo maiores níveis de desempenho quando um número maior de processadores era utilizado.

Outro trabalho relevante sobre medições de desempenho é apresentado em (POHL et al., 2004). Para esse caso foram realizadas avaliações de desempenho paralelo de uma

implementação tridimensional do MLB em arquiteturas Hitachi SR8000-F1, SGI Altix, NEC SX6, AMD Opteron e Intel Xeon. Nos testes, três aplicações foram desenvolvidas: uma para problemas de turbulência oriundas do cálculo da exaustão de partículas em veículos, outra para o cálculo da velocidade em torno de nano-partículas em aplicações da Engenharia Química e a última para a simulação de espumas metálicas. A análise dos resultados obtidos mostra bons índices de escalabilidade para todas as arquiteturas utilizadas em relação ao tamanho dos problemas simulados. Já o custo-benefício de processamento foi bem menor para as arquiteturas superescalares. Além disso, tais arquiteturas não dependem do fluxo dos dados, nem de previsões sobre a forma como o código será executado, assim como ocorre em arquiteturas vetoriais. Por outro lado, o ganho de desempenho das arquiteturas superescalares deixa muito a desejar em relação ao ganho obtido usando arquiteturas vetoriais.

Análise mais teóricas e completas da paralelização e de otimizações do método também foram feitas (KÖRNER et al., 2005). Além de caracterizar a estrutura do método, foram propostas diferentes estratégias de otimização do código seqüencial com a finalidade de aumentar a eficiência. Uma forma de otimizar as implementações do MLB é buscar uma boa disposição das estruturas de dados armazenadas na memória, as quais podem ser previamente definidas para o problema que está sendo tratado. Soluções desse tipo podem ser facilmente aproveitadas em qualquer tipo de implementação, inclusive em programas genéricos, desde que as otimizações mais específicas sejam deixadas de lado. Também é possível fazer o reaproveitamento de variáveis temporárias usadas como um espaço de troca entre os elementos do reticulado em cada interação. Nesse caso, porém, a descrição do código pode não ser tão clara. Em termos de paralelização, o caminho geralmente apresentado é o particionamento dos dados. Nesse caso, são descritos formas de comunicação e de balanceamento de carga eficientes, que fornecem um bom desempenho quando aplicados na prática.

Um outro fator relacionado ao desempenho do método refere-se aos aspectos físicos dos problemas simulados (ARTOLI; HOEKSTRA; SLOOT, 2005). Através do estudo das propriedades físicas é possível determinar estratégias de simulação com parâmetros otimizados. Com isso, resultados mais eficientes podem ser obtidos para cada caso avaliado, diminuindo simplesmente os cálculos que precisam ser feitos ou o número de iterações a ser utilizado. Trabalhos desse tipo já foram realizados para fluxos com movimentação variável, isto é, o canal por onde o fluxo passa possui uma dimensão variável, como é o caso dos vasos sangüíneos (ARTOLI; HOEKSTRA; SLOOT, 2005).

Em termos de particionamento de dados, diferentes soluções foram adotadas. Para uma estrutura reticular irregular foi proposto o método da Bisseção Ortogonal Recursiva - ORB (Orthogonal Recursive Bisection, em inglês) (KANDHAI et al., 1998). A idéia dessa estratégia está em dividir recursivamente o domínio dos dados, de maneira que cada processo tenha uma balanceamento de carga homogêneo. Ao invés de simplesmente distribuir os dados, o método leva em consideração o custo das operações, que pode ser maior para um determinada região do reticulado, caso esse tenha uma estrutura irregular. Tal abordagem foi testada para o MLB, apresentando bons índices de desempenho (PAN; PRINS; MILLER, 2004).

Uma outra técnica de particionamento utilizada é a Decomposição Regular de Domínio - RDD (do inglês, Regular Domain Decomposition) (DESPLAT; PAGONABARRAGA; BLADON, 2001). Nessa técnica os dados são distribuídos de maneira que cada processo opere sobre um mesmo volume de dados, sem se preocupar com a quantidade de operações, visto que as operações são geralmente as mesmas para cada dado. Essa abor-

dagem é perfeita para um sistema regular, visto que a melhor distribuição é obtida através de um processo de divisão simples. Embora a Decomposição Regular de Domínios já tenha sido utilizada na paralelização do MLB, não foram feitas maiores avaliações sobre as diferentes formas de decompor um reticulado (DESPLAT; PAGONABARRAGA; BLADON, 2001).

## 4.4 Conclusão: Posicionamento de Contribuição

Conforme visto nas seções anteriores, embora os recursos de programação de MPI apresentem todos os recursos necessários para o desenvolvimento de programas paralelos onde as estruturas de dados sejam particionadas em bloco, nenhum teste específico sobre os fatores que interferem na melhor estratégia foram feitas para o MLB. Assim, o principal foco deste trabalho é avaliar o uso de diferentes estratégias de particionamento de dados em bloco em implementações paralelas do MLB. Neste sentido, serão definidos os fatores que determinam as melhores formas de distribuição dos dados em estruturas regulares utilizadas para o armazenamento do reticulado.

# 5 IMPLEMENTAÇÃO PARALELA DO MÉTODO DE LATTICE BOLTZMANN

As estratégias de particionamento e distribuição de dados em blocos discutidas no capítulo anterior foram adotadas nesse trabalho como forma de diminuir o tempo de execução das implementações paralelas do MLB. O presente capítulo descreve os aspectos relacionados à adoção desses tipos de particionamento em implementações paralelas desenvolvidas para o método. Inicialmente, são apresentadas algumas idéias gerais utilizadas na implementação, tanto para o modelo bidimensional, quanto para o modelo tridimensional. Na seqüência, são detalhadas as estratégias de paralelização e os recursos de programação utilizados. Por fim, são feitas avaliações teóricas do impacto de diferentes formas de particionar os dados, comparando o tempo de comunicação consumido por cada estratégia em uma iteração.

## 5.1 Implementação

As implementações do MLB foram feitas para dois modelos de propagação de partículas. Para o caso bidimensional foi adotado o modelo D2Q9, enquanto que o modelo D3Q19 foi o escolhido para o caso tridimensional. A descrição desses modelos foi feita na Seção 3.2. A implementação das funções do método e do programa de testes foram escritas usando a linguagem de programação C padrão ANSI. Para ambos os casos foram adotadas três estruturas de dados, a fim de armazenar as informações utilizadas e manipuladas durante a execução do código. Tais estruturas definem, respectivamente:

1. as propriedades físicas;
2. a estrutura do reticulado;
3. parâmetros referentes a execução paralela.

A estrutura das propriedades físicas contém diversas informações. Tais informações envolvem valores como: densidade, aceleração (usado para determinar a velocidade inicial), escala do tempo de relaxação $\tau$ e o diâmetro do tubo real simulado usado para o cálculo do número de Reynolds. Esses valores são utilizados para iniciar algumas variáveis do reticulado, bem como para obter alguns valores macroscópicos dependentes dessas constantes após a simulação.

Para a estrutura de dados do reticulado são definidos:

- a dimensão do problema (2D ou 3D);
- o número de direções possíveis de deslocamento de cada ponto do reticulado;

- o número de pontos em cada uma das dimensões do reticulado completo e dos sub-reticulados a serem processados pelos nós;
- o posicionamento do sub-reticulado em relação ao reticulado completo inicial;
- uma matriz de posições que descreve a posição das barreiras e bordas do fluxo;
- uma região de memória alocada onde são armazenadas as informações físicas de cada um dos pontos do reticulado. O tamanho dessa região depende do número de direções de deslocamento e da dimensão do sub-reticulado.

Os parâmetros referentes à execução paralela englobam informações utilizadas no particionamento dos dados do reticulado. Tais parâmetros definem o número de divisões realizadas em cada dimensão do reticulado para a formação dos sub-reticulados, quais as sub-regiões que cada um dos processos irão operar e um identificador exclusivo para cada processo, o qual é utilizado nas comunicações cartesianas.

## 5.2 Paralelização

A paralelização do MLB está baseada em um modelo de programação *Single Program Multiple Data* (SPMD) (WILKINSON; ALLEN, 1998). Isso significa que existe um único código de programa que executa em cada processador, o qual atua sobre uma parcela distinta de dados em cada caso. Como a manipulação dos dados possui uma certa independência e isolamento entre diferentes subconjuntos de valores, tal forma de programação apresenta-se como uma das alternativas mais indicadas.

Uma vez escolhido o paralelismo de dados, é necessário aplicá-lo às implementações. O particionamento dos dados feito neste trabalho consiste em distribuir a estrutura do reticulado entre diversos processadores, com o intuito de realizar os cálculos independentemente sobre cada sub-região controlada por um processo. Diferente de abordagens que procuram dividir o problema apenas em uma dimensão, a implementação paralela proposta possibilita a divisão e distribuição dos sub-reticulados de várias formas. Assim, é possível dividir os dados em todas as dimensões coordenadas (2D ou 3D) pelos quais os modelos implementados são definidos, facilitando, assim, a escolha da estratégia mais eficiente para cada caso.

Uma das preocupações existentes na divisão dos dados é o balanceamento de carga. Como nas implementações paralelas do método a divisão dos dados é feita pelo número de processadores usados, é evidente que existirá um excedente quando a divisão for feita por um número de processadores não múltiplo do tamanho dos dados. Esse excedente precisará ser distribuído entre os processos já existentes. Para evitar que o excedente fique a cargo de um único processador ou que exista um processador com uma significativa quantidade de dados a menos a calcular em relação aos demais, optou-se por distribuir o excedente entre diversos processadores, de maneira que a diferença de carga entre cada processador seja de no máximo uma unidade em cada dimensão coordenada. Como exemplo, utilizando 6 processadores que dividem um reticulado em apenas uma determinada dimensão coordenada formada por 21 pontos, a distribuição do tamanho do sub-reticulado para cada processador teria, respectivamente, a seguinte ordem:

$$3,\ 3,\ 3,\ 4,\ 4\ e\ 4$$

Isso garante uma distribuição de dados mais igualitária entre os processadores. A diferença unitária é muito importante, uma vez que, considerando que haja uma divisão

irregular para um particionamento tridimensional, a diferença entre o melhor e pior casos será de apenas:

$$dif = (x+1) * (y+1) * (z+1) - x * y * z \tag{5.1}$$

Onde $dif$ é a diferença total de pontos, considerando as dimensões x, y e z do sub-reticulado.

Em virtude da propagação das informações dos pontos do fluido, existe uma dependência de dados entre as fronteiras dos sub-reticulados. Em cada iteração são realizados cálculos cujas informações precisam ser armazenadas e repassadas para outros processos. Assim, cada sub-reticulado possui um conjunto de pontos (borda) a mais, onde os valores a serem propagados são armazenados, conforme ilustrado para apenas o sub-reticulado 5 de um modelo bidimensional hipotético na Figura 5.1.

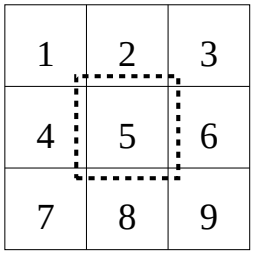


Figura 5.1: Bordas de armazenamento de valores temporários para sub-reticulados do modelo bidimensional

Em um segundo momento, após o armazenamento temporário dos resultados dos cálculos nas bordas, é feita a comunicação dos valores armazenados para os respectivos processos vizinhos de cada coordenada cartesiana. Dessa forma, os dados operados ficam constantes em todos os processos. Essa comunicação precisa ser feita tanto para os elementos vizinhos ortogonais, como para os diagonais, o que gera um número bastante grande de comunicações. Na Figura 5.2 são ilustrados, respectivamente, as comunicações necessárias para o caso bidimensional e tridimensional, onde as setas mais densas ilustram as comunicações ortogonais e as mais finas as comunicações diagonais.

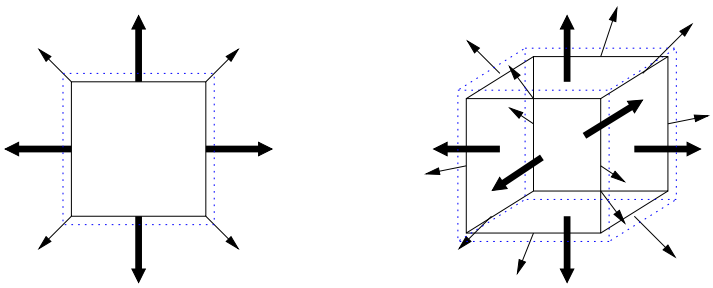

Figura 5.2: Fluxo de comunicação entre processos para a implementação bidimensional e tridimensional

A definição dos sub-reticulados e a troca de dados que ocorre entre os processadores durante as comunicações é realizada através dos recursos oferecidos pelas funções da biblioteca MPI. Os detalhes de programação referentes à comunicação são apresentados na próxima seção.

## 5.3 Funções de MPI Utilizadas na Implementação

Os recursos de programação utilizados de MPI facilitaram o particionamento dos dados do reticulado e a atualização dos dados pertinentes a cada um dos processos. Para tanto, foram utilizadas as seguintes diretivas:

- Mecanismos de mapeamento cartesiano;
- Funções de comunicação assíncronas;
- Funções de comunicação coletivas e síncronas.

Na implementação paralela do MLB, inicialmente é criada uma estrutura de comunicação cartesiana através da função `MPI_Cart_create()`, onde são definidos o número de dimensões e o número de partes em cada dimensão. Em seguida cada processo MPI é associado ao comunicador cartesiano através da função `MPI_Comm_rank()` e, a partir disso, mapeado via `MPI_Cart_coords()`, de maneira que cada processo esteja relacionado a uma única posição cartesiana. Desta forma, através do uso da função `MPI_Cart_rank()` é possível descobrir qual é o processo vizinho de uma determinada coordenada no momento em que uma comunicação irá ser efetuada.

Dependendo do tamanho do reticulado existe a possibilidade de que seja necessária a transmissão de uma grande quantidade de dados nas funções de atualização dos valores dos elementos das bordas. Como a diretiva de envio síncrono disponibilizada em MPI pela diretiva `MPI_Send()` pode ficar bloqueada até que uma mensagem tenha sido totalmente enviada, gerando esperas ativas, especialmente nos casos onde um grande volume de dados é manipulado, foi adotado um mecanismo de envio assíncrono não bloqueante. Esse recurso é oferecido pela função `MPI_Isend()`, garantindo a liberação do processo assim que a invocação do método é feita, sem a necessidade de espera para o envio dos dados. Já a recepção dos dados das bordas é feito com `MPI_Recv()`, uma função de recepção síncrona, que bloqueia o processo enquanto os dados não forem recebidos.

O cálculo paralelo da velocidade média do fluxo envolvido no método utiliza funções simples de comunicação. Visto que este cálculo utiliza apenas os valores de uma região do reticulado, nem todos os processos são considerados. Para realizar o cálculo da velocidade média todos os processadores que possuem os valores da região avaliada enviam suas médias parciais a um processador coordenador. Assim, foram utilizadas a função de envio bloqueante, `MPI_Send()`, e recebimento bloqueante, `MPI_Recv()`. Após a soma, o processador coordenador envia o valor da velocidade média a todos os processos através da função de comunicação coletiva `MPI_Bcast()`, gerando, assim, um ponto de sincronização.

## 5.4 Análise Teórica do Custo de Comunicação

A fim de avaliar o custo das operações de comunicação para as diferentes técnicas de particionamento adotadas foram desenvolvidas algumas análises teóricas. Tais análises consistem em mensurar o custo de comunicação de cada processador em cada iteração do método. Para tanto, foi adotado o modelo de execução paralela conhecido como **Modelo LogP** (CULLER et al., 1993, 1996) em virtude do mesmo ser mais realista e apropriado para sistemas de memória distribuída do que outras propostas (GIBBONS, 1989; VALIANT, 1990; CORREA et al., 2002);

O **Modelo LogP** é um modelo de máquina paralela que leva em conta explicitamente as características de desempenho da rede. Através dele é possível definir a granularidade das tarefas, uma vez que um número P finito de processadores é utilizado. Além do número de processadores P, o modelo tem como parâmetros a latência de comunicação L, a sobrecarga de empacotamento e tratamento das mensagens o, que deriva do termo em inglês *overhead*, e o intervalo mínimo entre o envio/recebimento das mensagens g, do inglês *gap*. Esse último parâmetro também é conhecido como contenção de rede, sendo equivalente ao inverso da vazão ($1/v_{out}$), sendo $v_{out}$ a vazão da rede. A Figura 5.3 ilustra a relação entre os componentes do modelo para um esquema de rede hipotético.

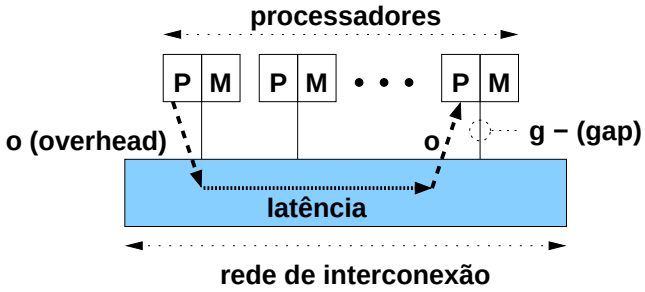


Figura 5.3: Esquema de funcionamento do modelo LogP (CULLER et al., 1996)

Para a simplificação das avaliações do custo de comunicação individual de cada processo realizadas neste trabalho foram considerados apenas a latência e a contenção da rede. Assim, o tempo de envio de uma mensagem de tamanho N pode ser sistematizado em:

$$tempo = L + Ng = L + N/v_{out} \tag{5.2}$$

### 5.4.1 Análise da Implementação Bidimensional

No modelo D2Q9, além da posição estática, existem 8 direções diferentes de deslocamento, sendo 4 ortogonais e 4 diagonais. Essas direções podem ser combinadas entre si de maneira que sejam definidos 4 faces ortogonais, a fim de simplificar o processo de comunicação dos dados que se encontram nas bordas dos sub-reticulados. Cada uma dessas 4 faces é composta de 3 direções de deslocamento: 1 direção ortogonal e 2 direções diagonais, conforme pode ser visto na Figura 5.4.

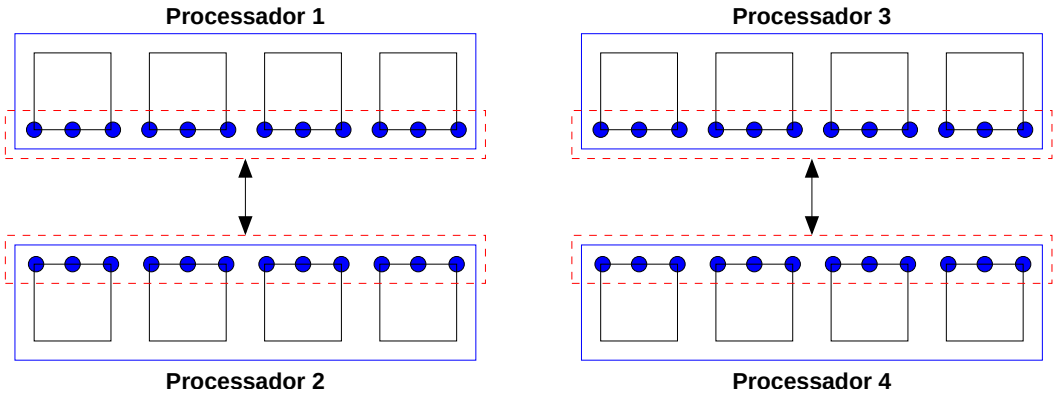


Figura 5.4: Exemplo de comunicação dos pontos das faces na implementação bidimensional

No exemplo acima, cada sub-reticulado é composto por 4 pontos. Para a simplificação

da ilustração são mostrados apenas as direções pertinentes a uma das faces. Já as linhas pontilhadas destacam as direções agrupadas que são comunicadas aos processos vizinhos. Considerando $tam_x$ e $tam_y$ a dimensão do reticulado em relação ao eixo x e y, respectivamente, e $proc_x$ e $proc_y$ o número de divisões feitas sobre o eixo x e y, respectivamente, então o tamanho das bordas dos sub-reticulados para as dimensões do eixo x e do eixo y, pode ser calculado por:

$$lado_x = tam_x / proc_x \tag{5.3}$$
$$lado_y = tam_y / proc_y \tag{5.4}$$

Como são 4 as bordas dos sub-reticulados, 2 em relação ao eixo x e 2 em relação ao eixo y, então o custo de comunicação será de:

$$custo\ ortogonal = 4L + 2(1/v_{out})(3(lado_x + lado_y)) \tag{5.5}$$

Caso uma das dimensões não tenha sido particionada, então tal dimensão não precisará comunicar seus dados.

Apesar da Figura 5.4 ilustrar a troca de dados entre apenas dois processos vizinhos para todas as direções de velocidade de uma determinada face do sub-reticulado, os valores das direções diagonais dos pontos extremos do sub-reticulado não devem ser considerados. Devido à direção de propagação da velocidade, tais valores precisam ser enviados aos processos diagonais vizinhos, conforme apresenta a Figura 5.5.

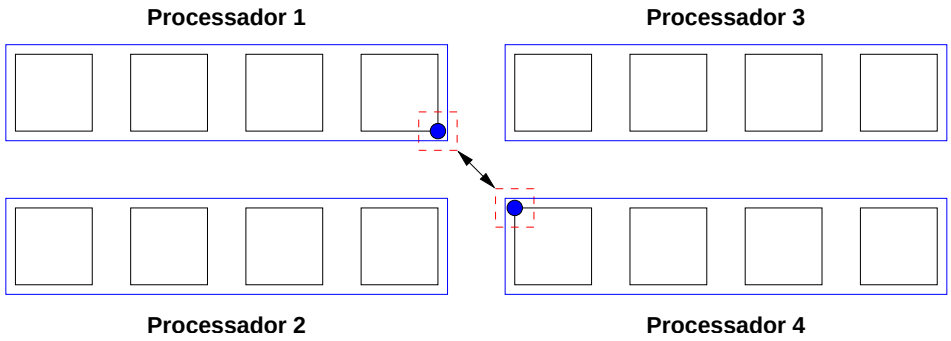


Figura 5.5: Exemplo de comunicação dos pontos diagonais na implementação bidimensional

Nesse caso, o Processador 1 envia as informações da direção diagonal do ponto mais extremo para seu vizinho diagonal, Processador 4. Para a simplificação foi mostrado apenas a comunicação entre esses dois processos. Todavia, o mesmo se repete entre as outras três diagonais. Por esse motivo, cada processo necessitará efetuar mais 4 comunicações de apenas um valor em cada caso, além das estimadas anteriormente:

$$custo\ diagonal = 4(L + (1/v_{out})) \tag{5.6}$$

Assim, o custo total da comunicação que um processo realiza será a soma das comunicações ortogonais com as comunicações diagonais, que no caso de um particionamento em duas dimensões será de:

$$custo\ total = 8L + 2(1/v_{out})(3(tam_y/proc_y + tam_x/proc_x) + 2) \tag{5.7}$$

### 5.4.2 Análise da Implementação Tridimensional

O cálculo de custo de comunicação do modelo tridimensional é semelhante ao modelo bidimensional. No modelo D3Q19 existem 18 direções diferentes de deslocamento além da posição estática. Essas 18 direções podem ser agrupadas entre si de maneira que sejam formadas 6 faces ortogonais. Cada uma delas é composta por 5 direções de deslocamento diferentes: uma direção ortogonal e 4 diagonais, conforme ilustra a Figura 5.6. Nessa figura foram exibidas apenas as direções de uma determinada face de dois sub-reticulados vizinhos.

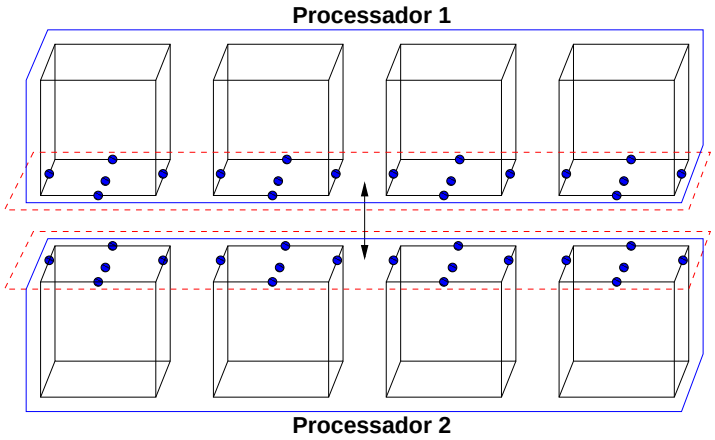


Figura 5.6: Exemplo de comunicação dos pontos das faces na implementação tridimensional

O número de pontos das faces denominadas de $face_x$, $face_y$, $face_z$, irá depender do tamanho das demais dimensões do sub-reticulado:

$$face_x = lado_y * lado_z \tag{5.8}$$
$$face_y = lado_x * lado_z \tag{5.9}$$
$$face_z = lado_x * lado_y \tag{5.10}$$

Onde o valor de $lado_x$, $lado_y$ e $lado_z$ irá depender da dimensão do reticulado e do número de divisões feitas sobre cada um dos eixos x, y e z, sendo calculados através de:

$$lado_x = tam_x/proc_x \tag{5.11}$$
$$lado_y = tam_y/proc_y \tag{5.12}$$
$$lado_z = tam_z/proc_z \tag{5.13}$$

Caso os lados de uma determinada face não tenham sido particionados, então tal face não precisá realizar a comunicação dos dados e conseqüentemente haverá um número menor de mensagens. Supondo, no entanto, que todas as faces tenham sido particionadas, então o custo de comunicação será de:

$$custo\ ortogonal = 6L + 2(1/v_{out})(5(face_x + face_y + face_z)) \tag{5.14}$$

Além da comunicação entre as faces, existe ainda a comunicação entre processos organizados na forma diagonal. Um exemplo de comunicação diagonal é mostrado na Figura 5.7. Novamente são utilizados dois processos de exemplo, exibindo a troca de dados em apenas uma das diagonais de propagação.

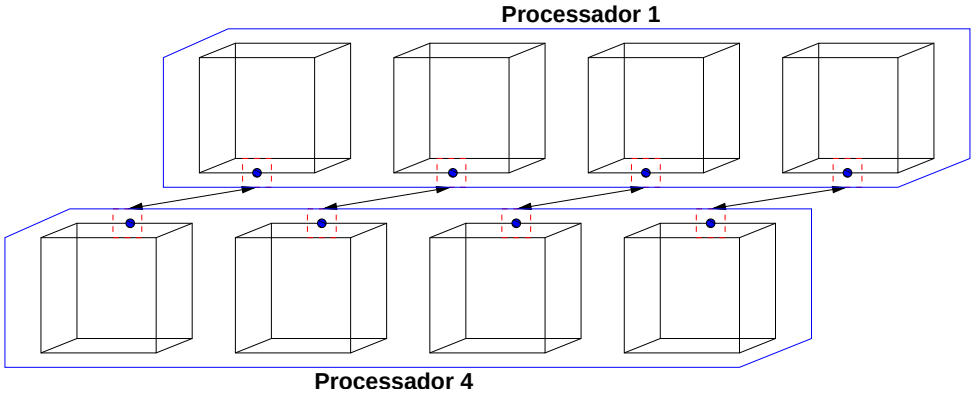


Figura 5.7: Exemplo de comunicação dos pontos diagonais na implementação tridimensional

No entanto, cada uma das 12 direções diagonais dos pontos que se encontram em uma das extremidades do sub-reticulado devem ser enviados aos seus respectivos processos diagonais vizinhos. Caso apenas uma das três dimensões tenha sido particionada, então haverá necessidade de comunicação de apenas 8 diagonais, perfazendo um custo de:

$$custo\ diagonal = 8L + 4(1/v_{out})(lado_1 + lado_2) \tag{5.15}$$

Sendo $lado_1$ e $lado_2$ o tamanho dos lados das dimensões que não foram particionadas. Entretanto, se o reticulado for particionado em duas ou três dimensões, então todas as 12 diagonais deverão ser enviadas. Para este caso, o custo de transmissão dos elementos diagonais será de:

$$custo\ diagonal = 12L + 4(1/v_{out})(lado_x + lado_y + lado_z) \tag{5.16}$$

onde, $tam_x$, $tam_y$ e $tam_z$ definem o número de elementos do sub-reticulado para a dimensão indicada e $proc_x$, $proc_y$ e $proc_z$ especificam o número de divisões feitas para a dimensão do índice subscrito. Da mesma forma que para a implementação bidimensional, o custo total de comunicação do caso tridimensional será obtido pela soma das comunicações das faces mais a soma dos custos de comunicação das diagonais, sendo este definido por:

$$custo\ total = 18L + (1/v_{out})(10(face_x + face_y + face_z) + 4(lado_x + lado_y + lado_z)) \tag{5.17}$$

### 5.4.3 Exemplos de Cálculo de Custos dos Particionamentos

As expressões desenvolvidas anteriormente foram utilizadas para avaliar o custo de comunicação de algumas configurações de particionamento. Os valores para a latência e vazão utilizados foram de, respectivamente, 42 $\mu$s (microssegundos) e 61,39 MB/s (Megabytes por segundo). A vazão foi obtida para um conjunto de dados transmitido de 128 KB (Kilobytes). Esses valores foram medidos no ambiente de testes descrito na Seção 6.3. O tipo de dados utilizado nas mensagens é de ponto flutuante e dupla precisão (tipo *double* em linguagem C), cujo tamanho é igual a 64 bits. A representação do particionamento e da distribuição dos dados dos sub-reticulados é feita através de uma notação "a X b" ou "a X b X c", onde a, b e c indicam, respectivamente, o número de partes ou dados particionados em relação aos eixos x, y e z.

#### 5.4.3.1 Um Exemplo para a Divisão Bidimensional

Para avaliar algumas configurações de particionamento do modelo bidimensional foi adotado um reticulado de 512 X 512 pontos. Este foi particionado entre 32 processos.

Dividindo-se o reticulado linearmente, 32 X 1 ou 1 X 32, o custo total de comunicação dos dados transmitidos por processo será em milissegundos (ms) equivalente a:

$$custo_A = 6L + 2(1/v_{out})(3 \times 512 \times 64 + 2 \times 64) = 3,19 \text{ ms} \quad (5.18)$$

Já para uma divisão em duas dimensões usando uma configuração 8 X 4, por exemplo, tem-se:

$$custo_B = 8L + 2(1/v_{out})(3(512 \times 64/4 + 512 \times 64/8) + 2 \times 64) = 1,44 \text{ ms} \quad (5.19)$$

Comparando-se os exemplos de cálculo de custo, $custo_A$ e $custo_B$, a diferença no tempo de comunicação do primeiro em relação ao segundo cai em mais da metade, o que valida o uso de particionamentos cartesianos para este caso. Uma vez que na divisão unidimensional o custo de comunicação é o mesmo para cada processador, independente do número de processadores utilizado, é possível obter sempre melhores resultados com o particionamento em blocos. Na teoria, isso significa que quanto maior o número particionamentos realizados em cada uma das duas dimensões, menor será o tempo necessário para a comunicação dos dados e maior será a diferença de tempo em relação ao particionamento linear. No entanto, na prática verifica-se um certo limite, que irá depender da latência da rede e da sobrecarga de particionamento.

#### 5.4.3.2 Um Exemplo para a Divisão Tridimensional

Para a divisão tridimensional foi considerado um reticulado de dimensão 128 X 128 X 128, o qual é dividido e distribuído entre 32 processos, gerando diferentes configurações de particionamento.

No caso do particionamento linear, como por exemplo 32 X 1 X 1, o custo de comunicação dos dados em cada iteração será de:

$$custo_C = 10L + (1/v_{out})(2 \times 5 \times 64 \times 128 \times 128 + 4 \times 64(128 + 128)) = 157,87 \text{ ms} \quad (5.20)$$

Considerando agora uma divisão bidimensional, como 1 X 4 X 8 partes, têm-se um tempo para a transmissão dos dados de:

$$custo_D = 16L + (1/v_{out})(10 \times 64 \times 128(128/4 + 128/8) + D) = 60,29 \text{ ms} \quad (5.21)$$

$$D = 4 \times 64(128 + 128/4 + 128/8) \quad (5.22)$$

Já para uma divisão tridimensional, tal como 2 X 4 X 4 partes, o custo das mensagens por iteração para realizar a comunicação será de:

$$custo_E = 18L + (1/v_{out})(2 \times 5 \times 64 \times E + 2 \times 64 \times 4 \times EE) = 50,47 \text{ ms} \quad (5.23)$$

$$E = (128/2)(128/4) + (128/4)(128/4) + (128/4)(128/2) \quad (5.24)$$

$$EE = 128/2 + 128/4 + 128/4 \quad (5.25)$$

Considerando o primeiro ($custo_C$) e o segundo ($custo_D$) casos de configuração de particionamento, pode-se observar uma redução em mais da metade do tempo necessário para a transmissão dos dados. Já entre o primeiro ($custo_C$) e o terceiro ($custo_E$) custo

de configuração de particionamento percebe-se que a última alternativa diminui o tempo em mais de $1/3$. Considerando que a aplicação paralela comunica uma grande quantia de dados em cada iteração, utilizar configurações de particionamento que atenuem tal procedimento é um ponto muito importante para melhorar a eficiência do paralelismo.

Apesar dos resultados teóricos serem muito bons para o particionamento em duas e três dimensões, é importante lembrar que a comunicação entre os processos ocorre par-a-par. Por esse motivo, os resultados práticos podem não ser tão bons em relação ao particionamento linear, visto que no particionamento em duas ou três dimensões cada processo depende de mais processos para realizar a sincronização. Com isso aumentam as chances de ocorrem atrasos e esperas no recebimento de mensagens.

# 6 RESULTADOS EXPERIMENTAIS: PARTICIONAMENTO LINEAR

As implementações paralelas do MLB descritas no capítulo anterior, além de analisadas teoricamente, também foram validadas empiricamente. Através da simulação de problemas reais foi possível observar um número maior de fatores que influenciam no desempenho das implementações paralelas do método. Neste sentido, este Capítulo apresenta a validação dos protótipos e os resultados paralelos obtidos através de algumas simulações, onde foram testadas diferentes configurações de particionamento e distribuição de dados lineares. A primeira parte do capítulo descreve o programa de testes utilizando o MLB. Na seqüência são relacionados os parâmetros e modelos de reticulado utilizados nos estudos de caso. O tópico seguinte descreve o ambiente de simulação e os critérios utilizados para a obtenção de valores experimentais. Por fim, a maior parte do capítulo apresenta, através de gráficos e tabelas, resultados físicos e computacionais obtidos através do uso de diferentes parâmetros de execução aplicados aos estudos de caso.

## 6.1 Descrição do Programa de Testes

A implementação desenvolvida para validar o código do MLB pode ser dividida em três partes, a saber:

- leitura de arquivos e preparação do reticulado;
- execução do método propriamente dito;
- cálculo de valores macroscópicos e gravação das informações em arquivos.

Inicialmente, o programa de testes faz a leitura de dois arquivos, contendo informações de entrada. O primeiro descreve os valores macroscópicos utilizados e algumas informações de configuração de execuções (tolerância, por exemplo). O segundo arquivo apresenta a estrutura do reticulado e a matriz de obstáculos. Nesse arquivo, a primeira linha contém o número de dimensões, o número de direções de deslocamento do reticulado, o número de elementos em cada dimensão e o número de pontos que formam os obstáculos. Baseado nessas informações iniciais é possível alocar a memória necessária para o armazenamento das estruturas. Na seqüência, a partir da segunda linha do arquivo, são obtidas as coordenadas dos pontos que definem os obstáculos e as paredes do tubo. Cada linha fornece a coordenada de um único obstáculo. Assim, será preciso ler tantas linhas, quantas o número de pontos de obstáculos indicar, a fim de obter a estrutura completa de bordas e barreiras. O protótipo de uma estrutura bidimensional desse tipo de arquivo é apresentado a seguir.

```
<dim> <dir> <lin> <col> <obs>
<x_1> <y_1>
<x_2> <y_2>
.
.
.
<x_obs> <y_obs>
```

Para os casos paralelos, cada processo lê do arquivo de descrição dos obstáculos apenas as coordenadas que estão contidas no seu subdomínio (sub-reticulado). Tal subdomínio é definido pelo tamanho do reticulado e pela configuração do particionamento, o qual é passado como argumento de entrada para a função de testes. Nessa parte da execução são inicializados, ainda, algumas constantes e a densidade individual de cada ponto do reticulado.

A execução do método propriamente dito ocorre conforme as operações definidas no algoritmo apresentado da Seção 3.8. Quando o programa é executado concorrentemente sobre os dados (uso de mais de um processador), é necessária a inserção de uma função responsável pela troca de dados (comunicação) logo após a função de propagação, a fim de que todos os processos operem sobre o conjunto de valores corretos. Já o critério de parada do laço principal é feito através da diferença entre as velocidades da iteração corrente com a iteração anterior, usando a média das velocidades dos pontos centrais do eixo $x$, conforme indicado na Figura 6.1. Desta forma, são utilizados todos os pontos que se concentram sobre a região indicada. Caso esses pontos se concentrem em processadores diferentes, a operação é realizada paralelamente. Uma alternativa, também utilizada nesse trabalho, é adotar um número fixo de iterações. Essa abordagem elimina o cálculo de velocidade e, conseqüentemente, a comunicação entre processos, caso esse fosse necessário para o cálculo paralelo da velocidade.

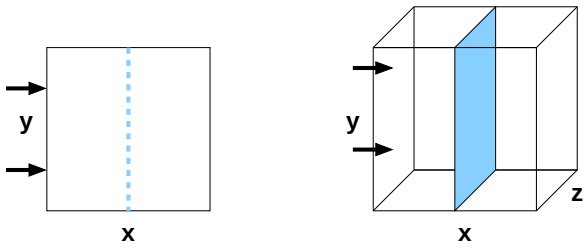


Figura 6.1: Uso dos pontos centrais do eixo $x$ para o cálculo da velocidade média

Para avaliar se o programa executa corretamente é feita a soma da densidade de todos os pontos, que deve permanecer constante durante a execução. Após as execuções do método são realizados o cálculo da velocidade média do fluxo e do número de Reynolds, os quais são gravados em um arquivo de saída, juntamente com o número de divisões realizados em cada direção coordenada, o número de iterações, a viscosidade e o tempo de execução. Em um outro arquivo são armazenados as coordenadas de todos os pontos, as velocidades médias das partículas para cada direção coordenada, a pressão e um marcador que indica se o ponto é um obstáculo ou não, tendo o arquivo o seguinte formato para o modelo tridimensional:

```
VARIABLES = X, Y, Z, VX, VY, VZ, PRESS, OBST
ZONE I=max_x, J=max_y, k=max_z F=POINT
```

```
x_1 y_1 z_1 vel_x_1 vel_y_1 vel_z_1 press_1 obst_1
x_2 y_2 z_2 vel_x_2 vel_y_2 vel_z_2 press_2 obst_2
.
.
.
x_n y_n z_n vel_x_n vel_y_n vel_z_n press_n obst_n
```

Essas informações podem ser utilizadas em um programa de visualização, tais como Grace, Gnuplot e TecPlot, a fim de avaliar graficamente o estado do fluxo após a simulação (TECPLOT, 2006).

## 6.2 Descrição dos Estudos de Caso

Como estudos de caso foram simulados a passagem de três fluxos de fluido através de canais com obstáculos. O objetivo dessas simulações foi avaliar o comportamento dos fluxos perante ambientes onde são encontradas barreiras rígidas, que exigem mudanças de direção no escoamento. Tais problemas foram criados artificialmente através de um programa gerador de obstáculos, que faz parte do programa de testes. Em todos os casos desenvolvidos, a propagação da força de entrada ocorre na direção formada pelo eixo $x$. A fim de facilitar a análise das influências das técnicas de paralelização adotadas foram escolhidos apenas formatos quadráticos ou cúbicos de reticulado.

O primeiro problema consiste de um fluxo bidimensional com tamanho de reticulado igual a 512 X 512 elementos. A distribuição dos obstáculos é composta de 5 barreiras dispostas ciclicamente ao longo do eixo $x$, conforme indicado na Figura 6.2. O tamanho de cada barreira é igual a metade do número de pontos que compõem os elementos da dimensão y. Já a distância entre cada uma das barreiras também é fixa, tendo-se utilizado para isso o valor de $1/5$ do tamanho total de pontos da dimensão x.

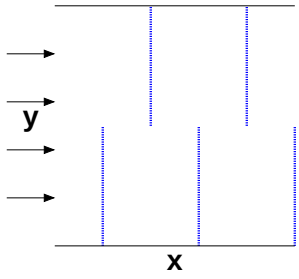


Figura 6.2: Disposição das barreiras no reticulado bidimensional

O segundo problema define um fluxo que cruza um canal tridimensional, cujo tamanho do reticulado é de 128 X 128 X 128 elementos. Para esse caso, existe uma única região retangular com obstáculos, a qual está disposta na parte anterior do eixo $x$, na posição definida por $1/3$ do tamanho total da dimensão x. Essa região engloba todos os pontos do eixo $y$ que se encontram na parte mais central em relação ao eixo $z$. A largura dessa parte central em torno do eixo $z$ possui uma extensão de $1/3$ do tamanho total de elementos que compõem essa dimensão. Graficamente o problema pode ser limitado conforme é exibido na Figura 6.3. Nessa figura, além dos obstáculos, são destacadas (listras) as faces que definem o canal, ficando duas faces abertas para a circulação do fluxo.

O terceiro estudo de caso é semelhante ao anterior em termos de formato e disposição dos obstáculos. A única diferença está no tamanho do reticulado, que possui uma dimensão de 256 X 256 X 256 pontos.

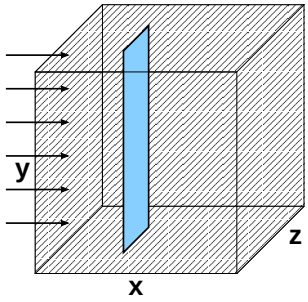


Figura 6.3: Disposição da barreira no reticulado tridimensional

Para todos os casos apresentados anteriormente, o valor das propriedades físicas iniciais do fluido simulado são os mesmos. As características do fluido adotado são: densidade de 0,1 $g/cm^3$, aceleração macroscópica $5 \cdot 10^{-4}\ cm^2/s$, fator $\tau$ de 1,85 e coeficiente linear de Reynolds igual a $10\ cm$.

## 6.3 Ambiente de Execução e Critérios para a Obtenção dos Resultados

O desempenho das implementações paralelas dos métodos foi medido nos *clusters labtec* e *corisco* do Instituto de Informática da UFRGS. O *cluster labtec* é composto por 20 máquinas duais Pentium III de 1.133 GHz e 512 KB de *cache* por processador, 1 GB de memória RAM por máquina e conexão de rede *Fast Ethernet*. O *cluster corisco* possui configuração semelhante, tendo 16 nós duais de capacidade de processamento de 1266 MHz e 512 KB de *cache* por processador, 512 MB de memória RAM por nó e duas opções de rede: *Fast Ethernet* e *Myrinet*. Quanto a parte de *software*, o sistema operacional adotado nos nós dos *clusters* é o sistema Linux, versão 2.6.8-3-686-smp, a versão do compilador GCC é a 3.3.5 e a distribuição do sistema operacional Linux instalado é Debian (versão 1:3.3.5-13). Nos testes dos códigos paralelos foi utilizado ainda a implementação *MPICH* versão 1.2.7 do padrão MPI.

Os resultados foram obtidos através da média de 10 execuções, sendo que a melhor e a pior execução foram excluídas. Desta forma, o desvio padrão para os valores que compõem as médias ficaram abaixo dos 2% em relação à média total. A opção por 10 execuções para cada caso testado se deve ao fato do desvio padrão ter ficado bastante baixo e também pelo fato das limitações existentes em termos de tempo de uso dos *clusters*. Uma vez que cada um dos casos testados consumia um tempo considerável de execução, o número de execuções precisou ser limitado.

Para testar as diferentes configurações de particionamento foram utilizados até 40 processadores (20 nós) do *cluster labtec*. A única exceção foi o teste com 64 processadores, em que foram adotados 32 processadores (16 nós) de cada *cluster*. Neste caso, é preciso observar que os valores de tempo obtidos precisam ser relativisados, uma vez que o conjunto de máquinas de cada *cluster* possui uma configuração diferente.

A captura dos tempos foi feita através da função *MPI_Wtime* da biblioteca MPI, que retorna em segundos o tempo transcorrido a partir de um determinado momento do passado. O tempo de execução foi obtido através da diferença do valor resultante da chamada de *MPI_Wtime* no início e no fim do laço principal do programa.

O critério de parada adotado para o modelo bidimensional foi um número predeterminado de iterações. Os testes feitos nesse modelo realizaram 15.000 iterações do laço

principal. Já para o modelo tridimensional, o critério de parada escolhido foi a variação de velocidade entre a iteração corrente e a iteração anterior da superfície central que se encontra no eixo $x$. Nos testes tridimensionais a diferença de velocidade escolhida foi de $10^{-4}\ cm/s$.

## 6.4 Resultados Obtidos para o Modelo Bidimensional

Os testes para o modelo bidimensional do MLB foram realizados usando como estudo de caso o reticulado de 512 X 512 pontos descrito na Seção 6.2.

### 6.4.1 Resultados Físicos

Após 150.000 iterações do laço principal do método, a velocidade média do fluido na coluna central do eixo $x$ foi de $1{,}35{\cdot}10^{-2}\ g/(cm{\cdot}s)$, a viscosidade foi de $1{,}54{\cdot}10^{-3}\ cm/s$ e o número de Reynolds obtido foi de 1,14. Para avaliar o processo de estabilização do fluxo durante a simulação, foram registrados ainda, a variação da velocidade média, conforme exibido na Figura 6.4. Através dessa figura, pode-se perceber que houve uma grande oscilação da velocidade média durante o transcorrer das iterações, até o fluxo atingir uma velocidade constante.

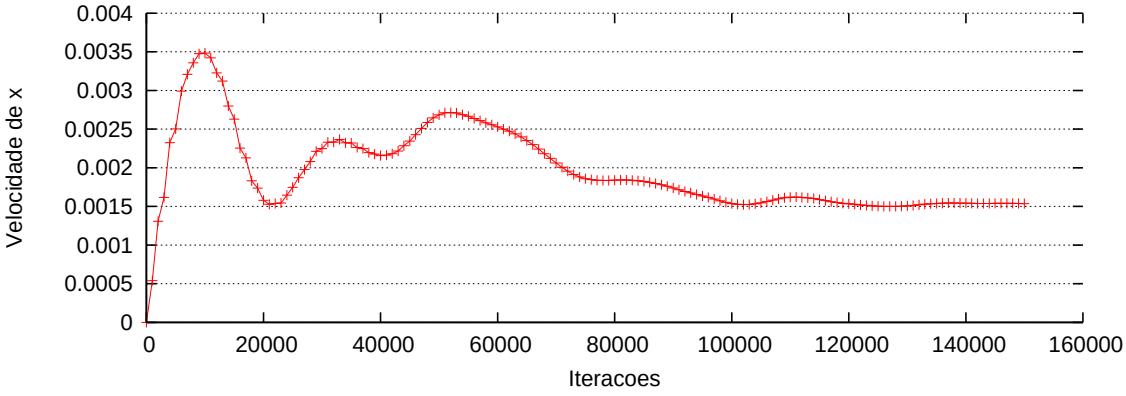


Figura 6.4: Evolução da velocidade média do fluxo durante a execução das iterações

Além da velocidade média no parte central do reticulado, foram extraídas, de algumas iterações, a distribuição de velocidade e pressão de todos os pontos do reticulado. Na Figura 6.5 são exibidos, respectivamente, a variação da velocidade em relação ao eixo $x$ em três momentos distintos: 90.000, 120.000 e 150.000 iterações, respectivamente de cima para baixo. Já na Figura 6.6 são apresentados os valores de pressão obtidos após 150.000 iterações do método. Todas as visualizações foram geradas através do *sofware* Tecplot.

Os resultados físicos obtidos possibilitam avaliar os efeitos decorrentes do uso de uma estrutura irregular sobre a circulação de um fluxo de fluido. Os resultados referentes à média de velocidade dos pontos (Figura 6.5) mostram que à medida que um número maior de iterações foi utilizado, mais coerente tornou-se o comportamento do fluxo perante a estrutura de distribuição dos obstáculos. A estrutura das barreiras fez com que a maior vazão do fluxo ocorresse na parte central do reticulado, através de um movimento senóide. Já nas laterais, entre as barreiras, existiram movimentos nulos ou zonas de refluxo. Através da Figura 6.6, pode-se perceber que existe uma maior pressão no fluido na parte anterior do fluxo, devido à ação das barreiras de contenção serem maiores no lado em que

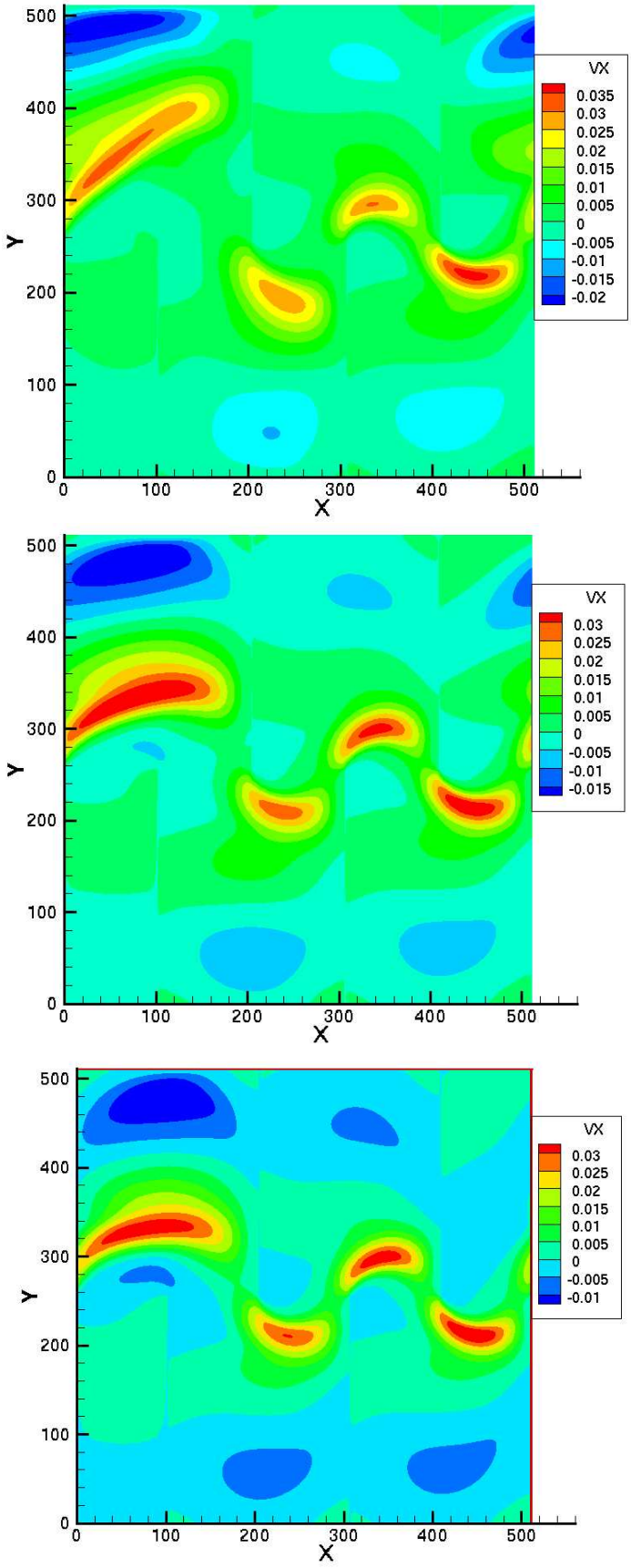


Figura 6.5: Visualização da velocidade em relação ao eixo $x$ do fluxo após 90.000, 120.000 e 150.000 iterações

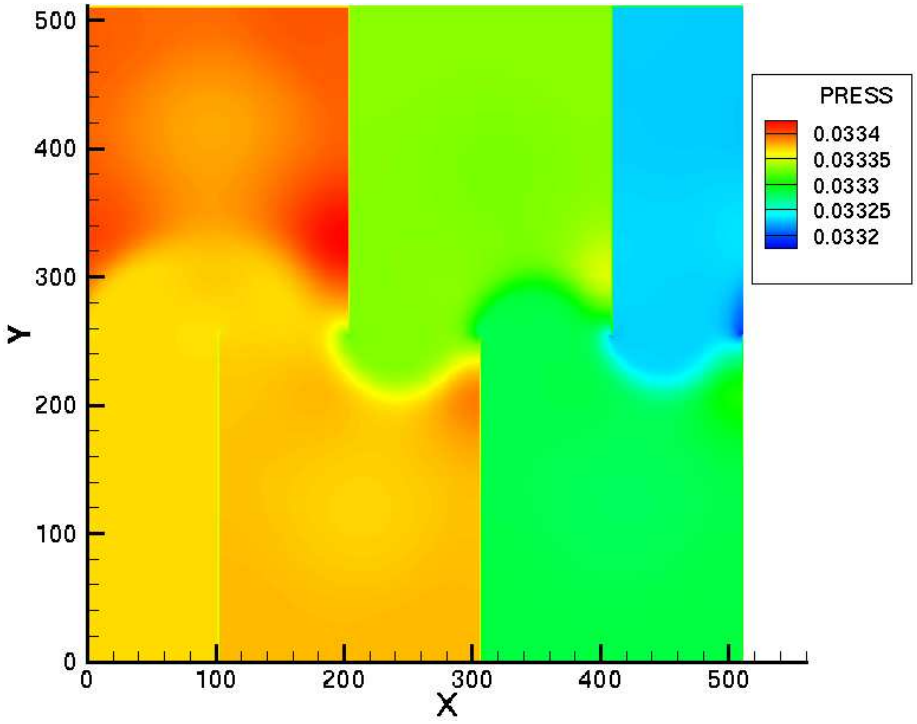


Figura 6.6: Visualização da pressão do fluxo após 150.000 iterações

a força de entrada é aplicada. Todavia, à medida que o fluxo dirigi-se para a parte posterior (saída), a pressão diminui entre as diferentes regiões do reticulado formadas pelos obstáculos.

### 6.4.2 Resultados Computacionais do Particionamento Unidimensional

Para avaliar a divisão unidimensional do modelo bidimensional do método foram feitas execuções utilizando até 40 processadores. Os tempos de execução obtidos estão apresentados na Figura 6.7, sendo que o tempo indicado é medido em segundos. Nessa figura o reticulado foi dividido em apenas uma das dimensões: divisão em linha (divisão paralela à dimensão x) ou divisão em coluna (divisão paralela à dimensão y), conforme indicado no gráfico. Para simplificar a visualização são mostrados apenas o tempo de execução seqüencial e os testes obtidos utilizando números pares de processadores.

Os ganhos de desempenho obtidos por ambas as formas de particionar linearmente o problema analisado são exibidos na Figura 6.8, sendo esses valores comparados com o desempenho linear ideal esperado (reta). Nesta dissertação utilizou-se o termo "ganho de desempenho" para o que também costuma ser chamado na literatura de "*speedup*" ou "fator de aceleração".

### 6.4.3 Análise dos Resultados Computacionais

As execuções paralelas do MLB apresentaram bons índices de desempenho. Independente da estratégia de particionamento adotada, as versões paralelas conseguiram reduzir significativamente os tempos de execução. Em relação ao particionamento linear pode-se observar um ganho de desempenho contínuo, tendo uma pequena oscilação a partir do momento em que a granularidade dos dados operados por cada processo se torna muito pequena. Assim, utilizando 40 processadores o ganho de desempenho do melhor caso foi de 27 vezes a execução seqüencial.

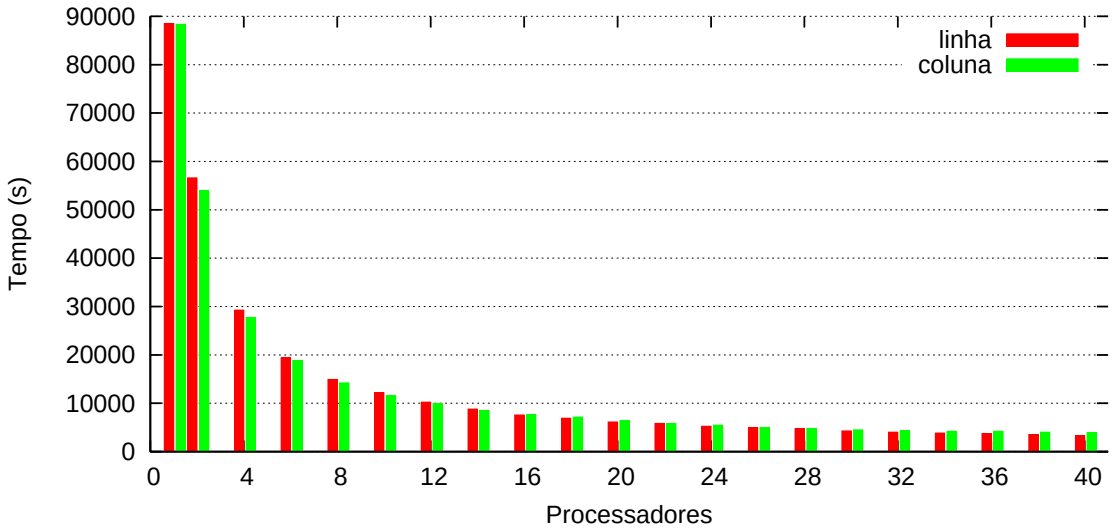


Figura 6.7: Tempo total de execução usando o particionamento unidimensional no modelo bidimensional

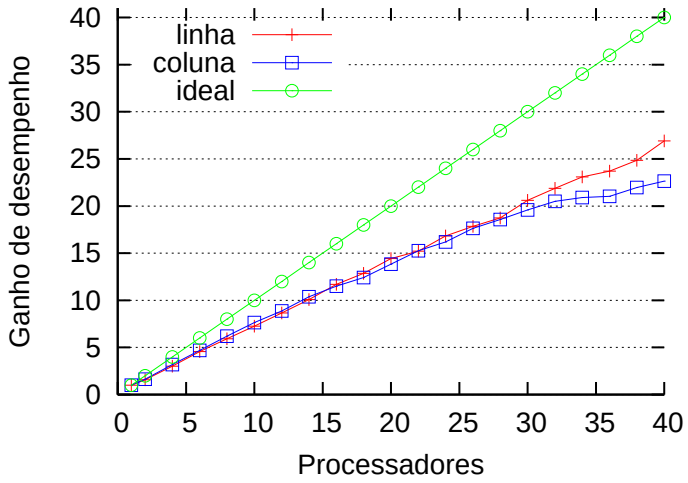


Figura 6.8: Ganho de desempenho usando o particionamento unidimensional no modelo bidimensional

Nos resultados apresentados, a melhor forma de particionamento para um determinado número de processadores acabou dependendo da maneira em que os dados foram acessados pelos processos. Assim, o particionamento em linhas (particionamento da direção do fluxo) obteve tempos menores para a maioria dos casos, uma vez que, através dessa estratégia, o acesso aos dados em memória e *cache* é feito de forma mais contínua, gerando menos falhas de acesso a *cache* e número de acessos a memória principal.

O tempo utilizando 2 processadores em relação as execuções seqüenciais diminuiu em torno de 60%. Esse valor não foi mais significativo devido ao sobrecusto gerado com o particionamento do problema e devido à contenção de memória existente pelo uso das capacidades máximas dos processadores de uma mesma máquina. Assim, o desempenho real não chega a ser tão alto como o desempenho linear esperado.

## 6.5 Resultados Obtidos para o Modelo Tridimensional

Para o modelo tridimensional foi utilizado os estudo de caso formado pelo reticulado de 128 X 128 X 128 elementos, conforme descrito na Seção 6.2.

### 6.5.1 Resultados Físicos

Para que os fluxos atingissem a estabilidade mínima esperada foram necessárias, respectivamente, 447 e 495 iterações do laço principal do MLB para os reticulados de dimensão 128 X 128 X 128 e 256 X 256 X 256. Em ambos os casos o valor da viscosidade foi o mesmo, 1,35 $\cdot 10^{-2}$ $g/(cm \cdot s)$, uma vez que esse valor é determinado apenas pelas condições de entrada. Para o reticulado de 128 X 128 X 128 elementos a velocidade média do fluxo na região central do eixo $x$ após 447 iterações foi de 6,12 $\cdot 10^{-2}$ $cm/s$ e o número de Reynolds foi igual a 45,26. Já para o reticulado de 256 X 256 X 256 a velocidade média foi igual a 7,04$\cdot 10^{-2} cm/s$ e o número de Reynolds foi de 52,07.

Para os resultados obtidos pelo reticulado de dimensão 128 X 128 X 128 foram geradas algumas imagens referentes às propriedades físicas do fluxo. Na Figura 6.9 e Figura 6.10 são apresentadas respectivamente a velocidade e a pressão do fluxo para as regiões formadas pelos pontos 32, 64 e 96 do eixo $x$. A escolha dessas regiões tem como objetivo oferecer uma visão geral sobre a distribuição das propriedades físicas em todo o fluido. Cada uma dessas regiões representa, portanto, uma fatia do reticulado sob o ponto x indicado.

Analisando-se as ilustrações da Figura 6.9 pode-se observar o efeito da disposição da placa sob o fluxo do fluido. Na parte mais central e anterior do reticulado o fluxo permanece estático em função da ação da barreira. Já para os pontos dispostos após a barreira, segundo à direção do fluxo, o efeito dos obstáculos sobre à propagação do fluxo começa a diminuir à medida que os pontos se afastam do mesmo. O mesmo ocorre com a distribuição da pressão, conforme a Figura 6.10, a qual é maior na região central anterior do fluxo. Após o fluxo passar pela barreira, ele lentamente vai estabilizando a distribuição da pressão.

De forma semelhante à Figura 6.9, a Figura 6.11 apresenta a velocidade sob os pontos 42, 43 e 44 do eixo $x$. A escolha dessa região serve para avaliar o comportamento do fluxo em torno do obstáculo, visto que a barreira está disposta sob o ponto 43 do eixo $x$ do reticulado. A distribuição das velocidades para essa parte do reticulado, mostra que existe um refluxo na parte central do fluxo, enquanto que nas laterais abertas ocorre a passagem do fluxo em uma maior velocidade, principalmente na região anterior do obstáculo (primeira ilustração). As velocidades nulas na parte central da segunda ilustração, não são a velocidade em si, mas uma representação da barreira.

### 6.5.2 Resultados Computacionais do Particionamento Unidimensional

A Figura 6.12 apresenta os tempos de execução obtidos utilizando até 40 processadores. Com exceção da execução seqüencial, apenas os resultados obtidos com números de processadores pares foram ilustrados. Nesse teste, o reticulado foi dividido nas dimensões x, y ou z, conforme indicado respectivamente no gráfico.

O ganho de desempenho das execuções paralelas para o caso analisado é exibido na Figura 6.13. Para simplificar a avaliação foram usadas somente os resultados dos particionamentos feitos sob o eixo $y$, sendo esses valores comparados com um desempenho linear ideal.

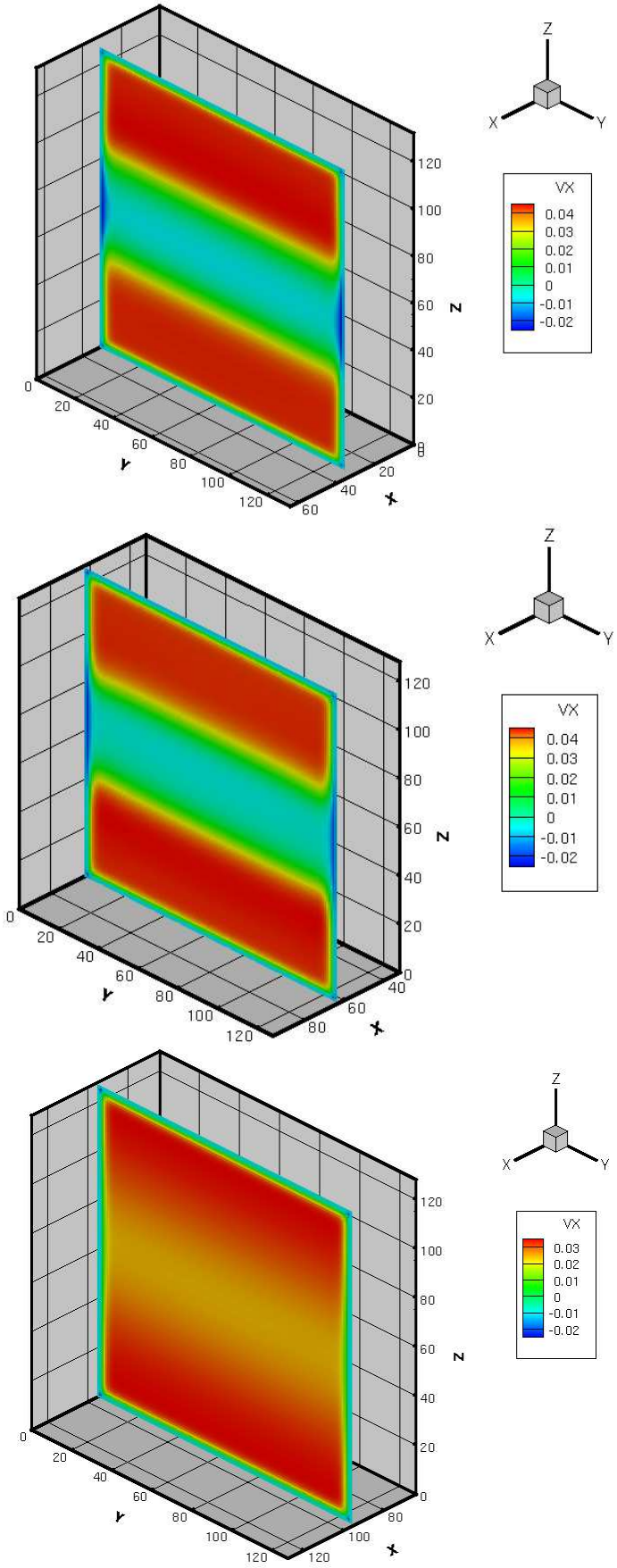


Figura 6.9: Velocidade do fluxo nas regiões formadas pelos pontos 32, 64 e 96 do eixo $x$

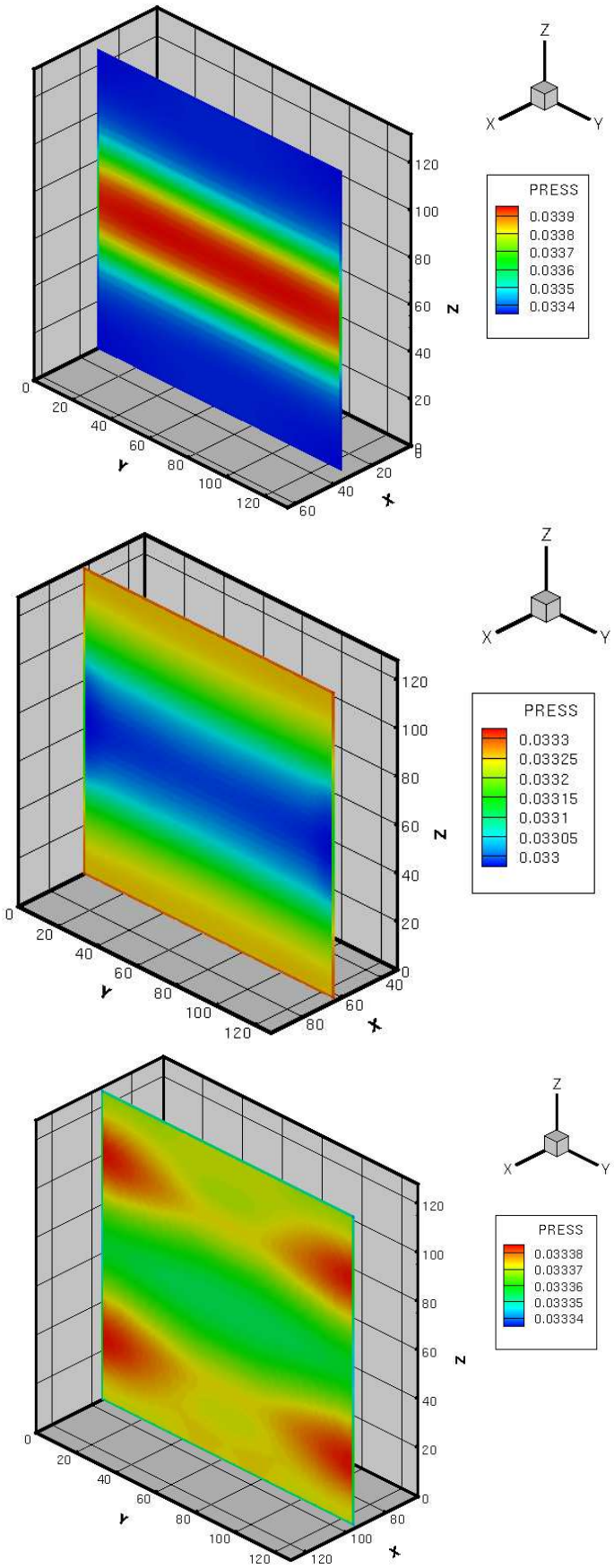


Figura 6.10: Distribuição da pressão do fluxo nas regiões formadas pelos pontos 32, 64 e 96 do eixo $x$

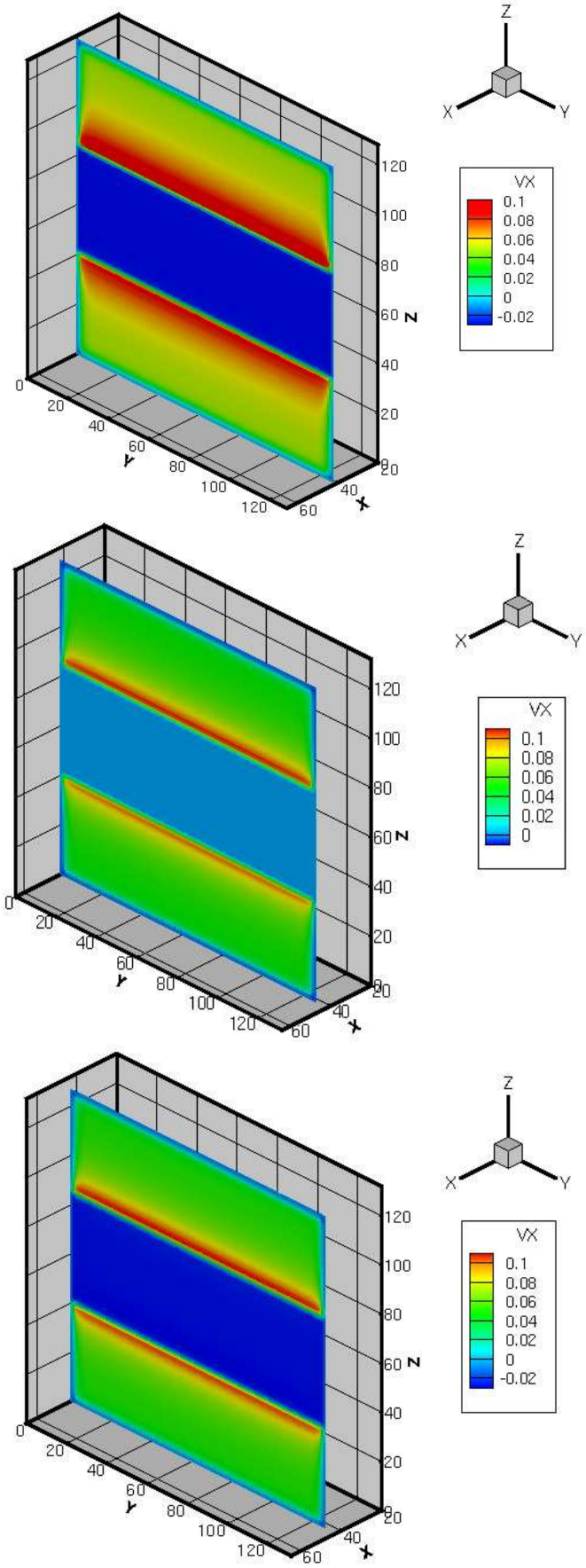


Figura 6.11: Velocidade do fluxo nas regiões formadas pelos pontos 42, 43 e 44 do eixo $x$

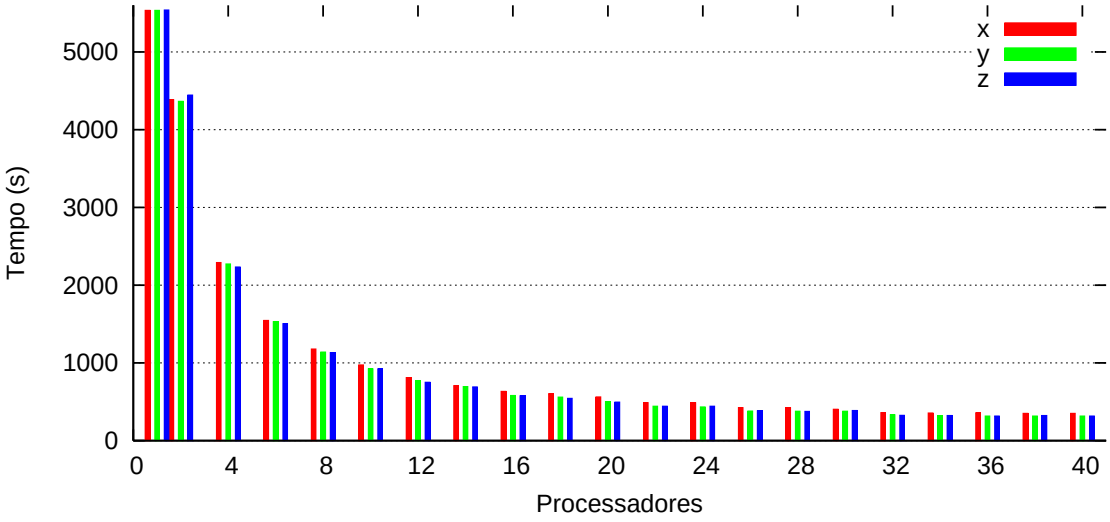


Figura 6.12: Tempo total de execução usando o particionamento unidimensional no modelo tridimensional

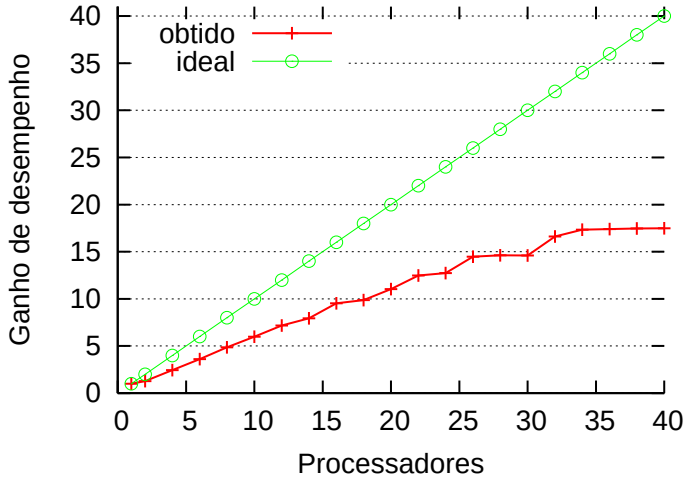


Figura 6.13: Ganho de desempenho usando o particionamento unidimensional no modelo tridimensional

### 6.5.3 Análise dos Resultados Computacionais

Os resultados obtidos na divisão unidimensional da implementação do modelo 3D do MLB, semelhantemente aos encontrados no modelo 2D, apresentaram sucessivos aumentos de desempenho à medida que um número maior de processadores foi utilizado. Devido ao pequeno tamanho do problema o ganho de desempenho máximo obtido usando 40 processadores para o melhor caso foi de algo em torno de 18 vezes a execução seqüencial, o que não chega a ser um valor expressivo de ganho de desempenho.

Em relação às formas de particionamento, a divisão do eixo $x$ não apresentou desempenhos tão bons, uma vez que tal estratégia possui um sobrecusto oriundo do cálculo da velocidade média em cada iteração em paralelo. Como esse cálculo é feito sobre a região central do eixo $x$, quando o reticulado estiver particionado nesse sentido, haverá a necessidade de comunicação para obter o valor da média. Uma exceção ao pior desempenho particionando o reticulado no eixo $x$ ocorre quando dois processadores são utilizados. Nesse caso, devido à questões de distribuição de dados e contenção de memória, o pior

tempo de execução ficou com o particionamento do eixo $z$.

# 7 RESULTADOS EXPERIMENTAIS: PARTICIONAMENTO EM BLOCOS

No capítulo anterior foram apresentados os resultados obtidos utilizando estratégias de particionamento lineares. Embora seja uma abordagem simples, para a maioria dos casos houve uma significativa redução dos tempos de execução à medida que um número maior de processadores era utilizado. No entanto, isso não limita a capacidade de aumentar o desempenho das implementações paralelas. Nesse sentido, este capítulo apresenta os resultados obtidos utilizando estratégias de particionamento em blocos. Como estudo de caso foram utilizados os mesmos reticulados utilizados para validar o particionamento linear. A divisão do capítulo consiste essencialmente dos resultados obtidos para o modelo 2D e do modelo 3D do método. Em cada caso, são apresentadas ainda, algumas análises referentes aos valores obtidos.

## 7.1 Resultados Obtidos para o Modelo Bidimensional

### 7.1.1 Resultados Computacionais do Particionamento Multidimensional

A divisão em múltiplas dimensões foi realizada particionando o reticulado de 512 X 512 pontos em diferentes configurações possíveis, usando 32, 36 e 40 processadores. Os tempos de execução obtidos para esses casos são apresentados, respectivamente na Figura 7.1, Figura 7.2 e Figura 7.3. Uma análise quantitativa é feita na Tabela 7.1, onde são apresentados os ganhos de desempenho reais entre o particionamento em uma e duas dimensões dos melhores casos.

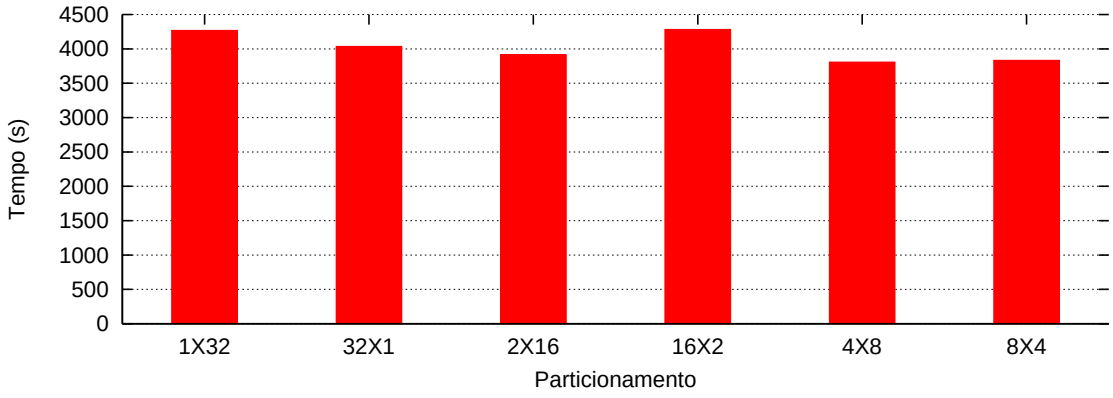


Figura 7.1: Tempo de execução obtido utilizando 32 processadores para diferentes configurações de particionamento do modelo bidimensional

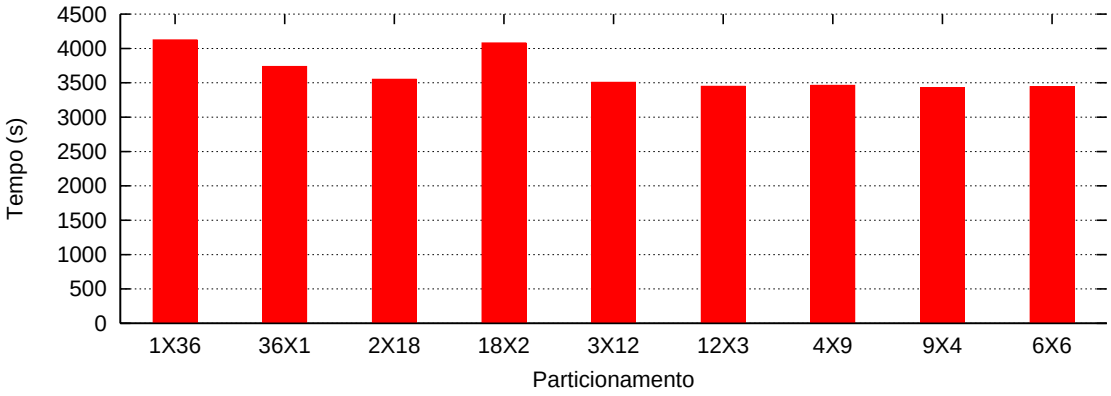


Figura 7.2: Tempo de execução obtido utilizando 36 processadores para diferentes configurações de particionamento do modelo bidimensional

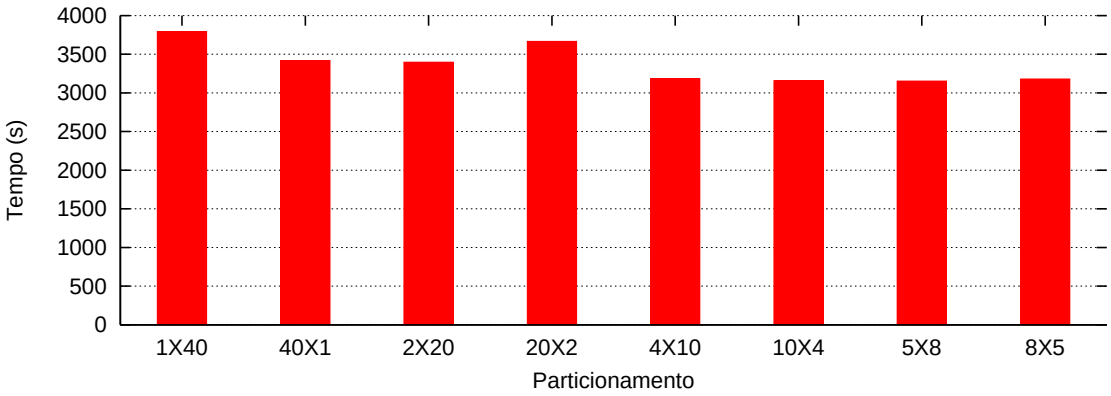


Figura 7.3: Tempo de execução obtido utilizando 40 processadores para diferentes configurações de particionamento do modelo bidimensional

Tabela 7.1: Relação entre os melhores tempos de execução usando 32, 36 e 40 processadores para as divisões em 1 e 2 dimensões

| **Distribuição** | Tempo | % de ganho em relação a divisão unidimensional |
|---|---|---|
| 32X1 | 4037,49 | - |
| 4X8 | 3808,57 | 5,67% |
| 36X1 | 3738.54 | - |
| 9X4 | 3430,8 | 8,23% |
| 40X1 | 3418,07 | - |
| 5X8 | 3152,42 | 7,77% |

### 7.1.2 Análise dos Resultados Computacionais

A divisão em múltiplas dimensões apresentou um comportamento semelhante entre os resultados obtidos usando 32, 36 e 40 processadores. Através desses resultados pode-se observar uma diminuição do tempo total de execução do particionamento em blocos em relação ao particionamento linear, tanto na divisão em linha como na divisão em coluna. Esses resultados apresentaram-se melhores para os particionamentos cujo tamanho dos sub-reticulados são mais quadrados. As comparações entre os melhores tempos de exe-

cução paralela usando a divisão em blocos apresentaram reduções de até 8% em relação ao melhor tempo obtido na divisão linear. Tais valores são expressivos, uma vez o tempo de processamento do método é significativamente maior que o tempo de comunicação.

Apesar do particionamento em bloco teoricamente diminuir o custo de comunicação, existem alguns casos onde o tempo de execução usando o particionamento em blocos foi um pouco pior que o particionamento linear. Isso aconteceu sempre que o reticulado foi dividido em duas colunas, independente do número de linhas adotado no particionamento. A razão para isso ocorrer está ligada ao mecanismo de comunicação. No processo de comunicação são realizadas primeiramente a troca de dados entre os lados divididos em colunas (eixo x) e, posteriormente, entre os divididos em linhas (eixo y). Como no caso analisado os processos são mais dependentes dos vizinhos do eixo y, possíveis atrasos na realização da transmissão de dados no eixo x são responsáveis por aumentar o tempo de espera no processo de sincronização do eixo y. No caso do reticulado ser dividido em duas linhas e em um número qualquer de colunas, não ocorre esse problema de retardo, visto que as comunicações entre as colunas é feita primeiro.

## 7.2 Resultados Obtidos para o Modelo Tridimensional

### 7.2.1 Resultados Computacionais do Particionamento Multidimensional

A divisão em múltiplas dimensões para um reticulado de 128 X 128 X 128 pontos foi feita com diferentes combinações de particionamento usando 27, 32 e 40 processadores.

Para o primeiro caso foi escolhido um número de processadores que permitisse a divisão das três dimensões por um mesmo denominador comum. Por isso, foram utilizados 27 processadores. Os tempos de execução, tanto do particionamento linear, como o realizado em duas ou três dimensões, são ilustrados na Figura 7.4, enquanto que na Tabela 7.2 são comparados quantitativamente os melhores tempos de execução de cada forma de particionamento.

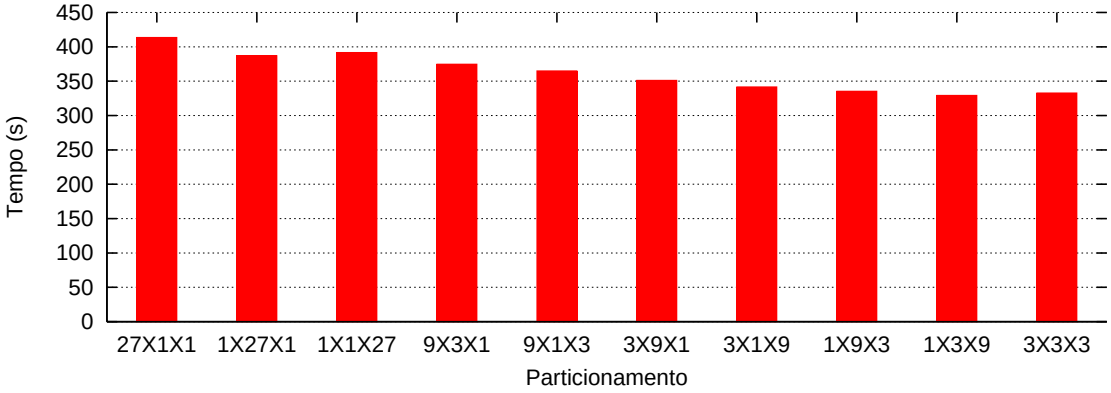


Figura 7.4: Tempo de execução obtido utilizando 27 processadores para diferentes configurações de particionamento do modelo tridimensional

No segundo caso, os resultados foram obtidos dividindo-se o reticulado entre 32 processadores, conforme apresentados nas Figura 7.5 e Figura 7.6, onde são mostrados respectivamente os tempos de execução para o particionamento do reticulado em duas e em três dimensões. Já a Tabela 7.3 faz uma comparação entre os tempos obtidos para os melhores casos da divisão em uma, duas e três dimensões.

O terceiro caso escolhido, no qual são utilizados 40 processadores, segue a estra-

Tabela 7.2: Relação entre os melhores tempos de execução usando 27 processadores para as divisões em 1, 2 e 3 dimensões

| **Distribuição** | Tempo | % de ganho em relação a divisão unidimensional |
|---|---|---|
| 1X27X1 | 387,13 | - |
| 1X3X9 | 329,26 | 14,95% |
| 3X3X3 | 332,8 | 14,04% |

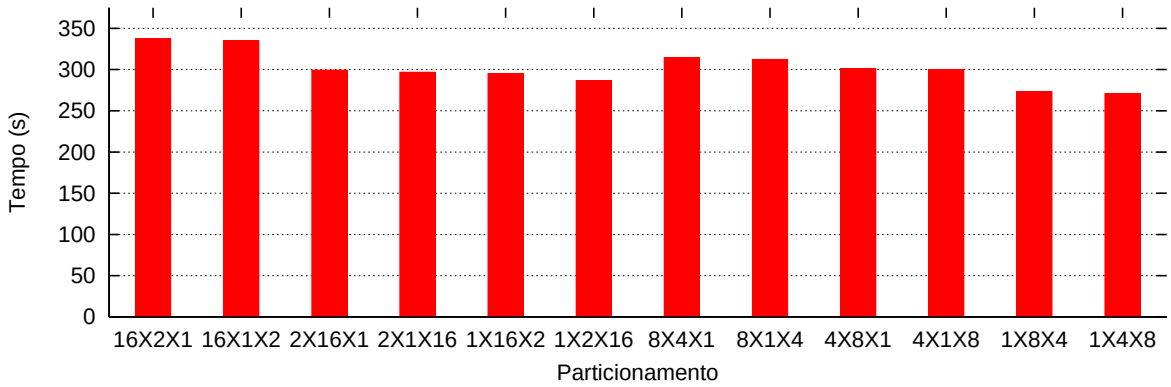


Figura 7.5: Tempo de execução obtido utilizando 32 processadores para diferentes configurações de particionamento em 2 dimensões do modelo tridimensional

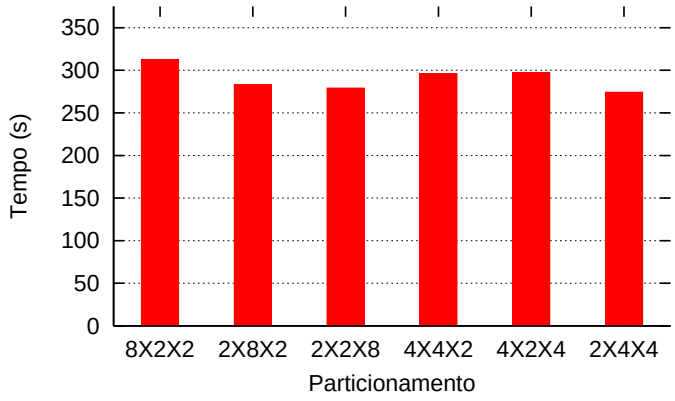


Figura 7.6: Tempo de execução obtido utilizando 32 processadores para diferentes configurações de particionamento em 3 dimensões do modelo tridimensional

Tabela 7.3: Relação entre os melhores tempos de execução usando 32 processadores para as divisões em 1, 2 e 3 dimensões

| **Distribuição** | Tempo | % de ganho em relação a divisão unidimensional |
|---|---|---|
| 1X1X32 | 329,99 | - |
| 1X4X8 | 274,18 | 15,44% |
| 2X4X4 | 270,52 | 18,02% |

tégia de análise adotada anteriormente, comparando na Figura 7.7, na Figura 7.8 e na Tabela 7.4, respectivamente, os tempos obtidos para as execuções com particionamento em duas dimensões, três dimensões e os valores obtidos para os melhores tempos para cada tipo de particionamento.

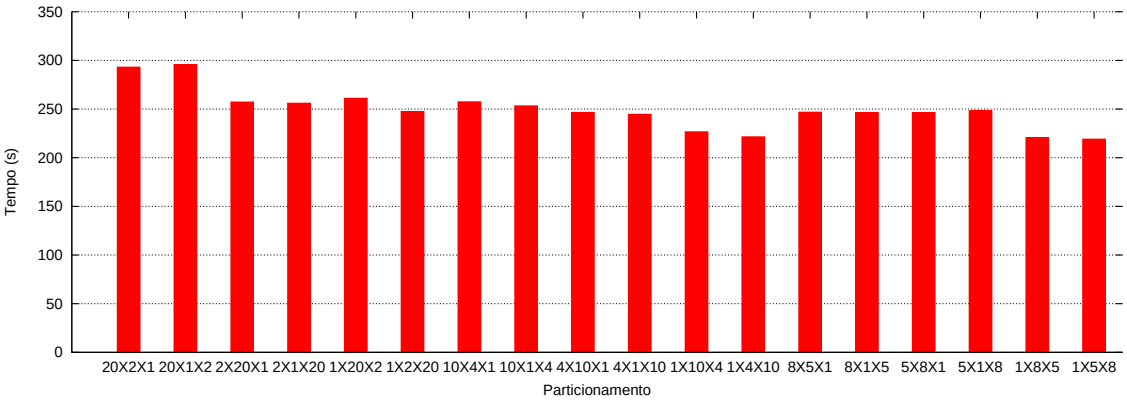


Figura 7.7: Tempo de execução obtido utilizando 40 processadores para diferentes configurações de particionamento em 2 dimensões do modelo tridimensional

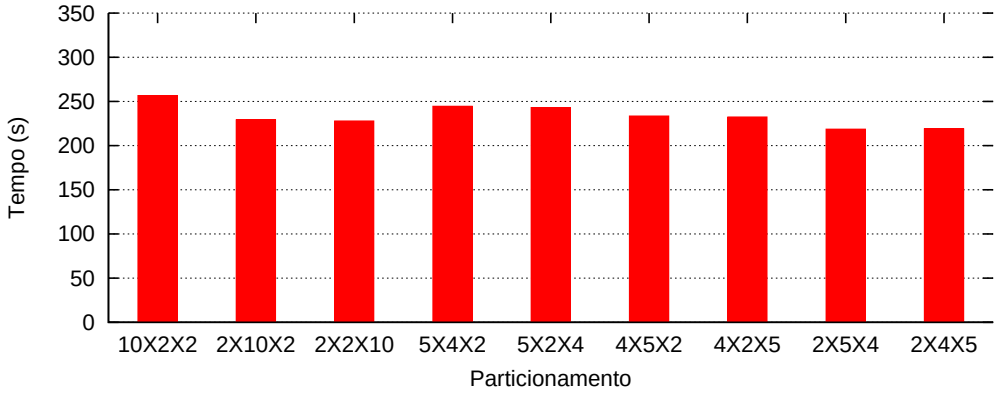


Figura 7.8: Tempo de execução obtido utilizando 40 processadores para diferentes configurações de particionamento em 3 dimensões do modelo tridimensional

Tabela 7.4: Relação entre os melhores tempos de execução usando 40 processadores para as divisões em 1, 2 e 3 dimensões

| **Distribuição** | Tempo | % de ganho em relação a divisão unidimensional |
|---|---|---|
| 1X40X1 | 318,56 | - |
| 1X5X8 | 219,07 | 31,23% |
| 2X5X4 | 218,63 | 31,37% |

Escolhendo-se os menores tempos de execução obtidos pelas diferentes configurações de particionamento para cada número de processadores usado, tanto a partir dos casos apresentados anteriormente, como os valores obtidos para os demais casos, é possível obter uma nova relação de ganho de desempenho. A Figura 7.9 apresenta o ganho

de desempenho considerando somente os menores tempos obtidos para cada configuração de particionamento de um determinado número de processadores. Tais valores são comparados com o ganho de desempenho ideal esperado, o qual é exibido através da reta crescente.

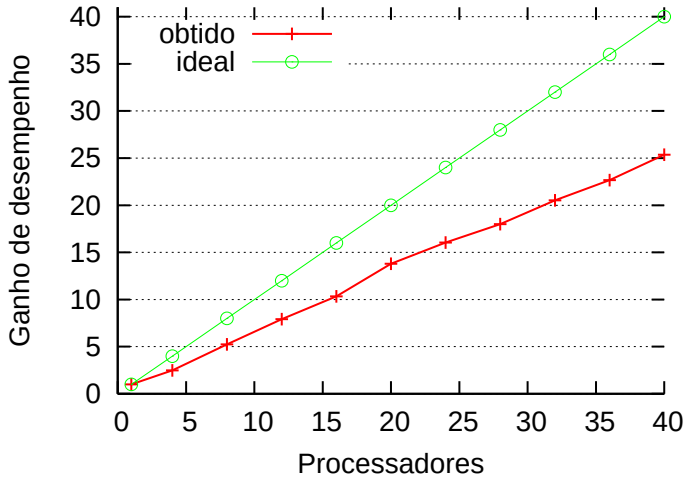


Figura 7.9: Ganho de desempenho utilizando os menores tempos paralelos de execução obtidos pelas diferentes configurações de particionamento do modelo tridimensional

Além do problema testado anteriormente foi utilizado também um reticulado de 256 X 256 X 256 pontos, a fim de avaliar o desempenho paralelo em problemas maiores. Para este caso, o reticulado foi particionado em diferentes configurações utilizando 32 e 64 processadores. As Figuras 7.10 e 7.11 apresentam, respectivamente, os tempos de execução particionando o reticulado entre 32 processos em 2 e 3 dimensões. Já na Tabela 7.5 são relacionados os melhores ganhos de desempenho para o particionamento em múltiplas dimensões em relação ao melhor tempo de execução utilizando o particionamento unidimensional. Os tempos de execução para o particionamento em 2 e 3 dimensões entre 64 processos são mostrados nas Figuras 7.12 e 7.13. Uma comparação entre os piores e melhores tempos de execução usando 32 e 64 processadores é feita na Tabela 7.6 para avaliar a escalabilidade da implementação tridimensional.

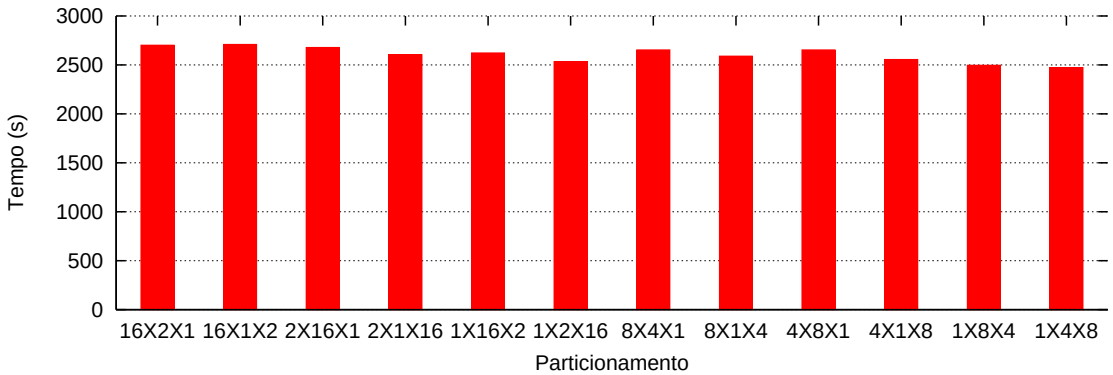


Figura 7.10: Tempo de execução obtido utilizando 32 processadores para diferentes configurações de particionamento em 2 dimensões do modelo tridimensional para um reticulado de 256 X 256 X 256 pontos

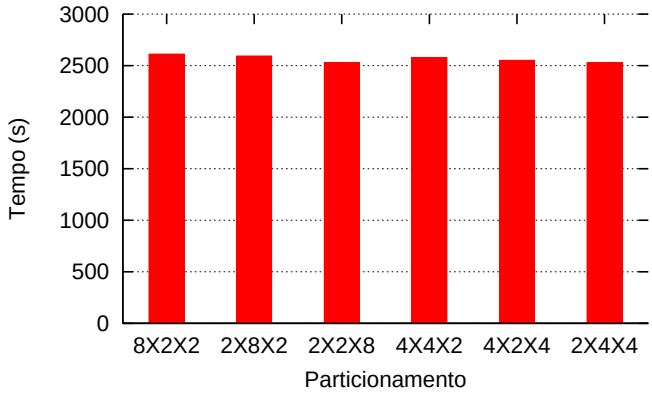


Figura 7.11: Tempo de execução obtido utilizando 32 processadores para diferentes configurações de particionamento em 3 dimensões do modelo tridimensional para um reticulado de 256 X 256 X 256 pontos

Tabela 7.5: Relação entre os melhores tempos de execução usando 32 processadores para as divisões em 1, 2 e 3 dimensões para um reticulado de 256 X 256 X 256 pontos

| **Distribuição** | Tempo | % de ganho em relação a divisão unidimensional |
|---|---|---|
| 1X1X32 | 2645,24 | - |
| 1X4X8 | 2473,78 | 6,48% |
| 2X4X4 | 2527,33 | 4,46% |

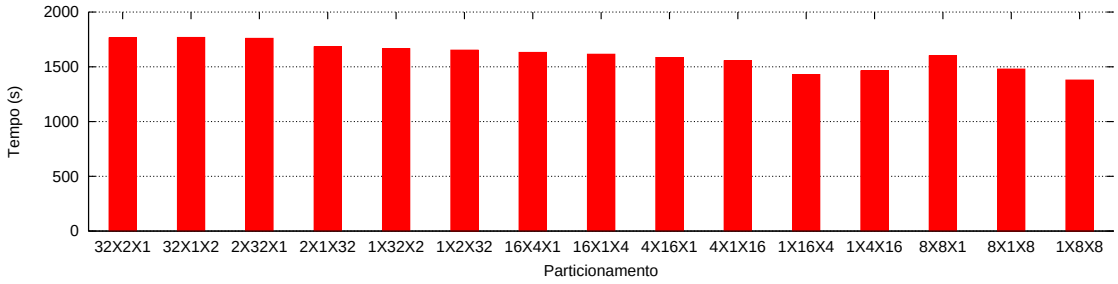


Figura 7.12: Tempo de execução obtido utilizando 64 processadores para diferentes configurações de particionamento em 2 dimensões do modelo tridimensional para um reticulado de 256 X 256 X 256 pontos

Tabela 7.6: Relação entre os melhores e piores tempos de execução usando 32 e 64 processadores para as divisões em 2 e 3 dimensões de um reticulado de 256 X 256 X 256 pontos

| | 32 processadores | | 64 processadores | | |
|---|---|---|---|---|---|
| **Caso** | Distrib. | Tempo | Distrib. | Tempo | Redução |
| Pior 2 dimensões | 16X1X2 | 2709.57 | 32X1X2 | 1768.39 | 34,74% |
| Melhor 2 dimensões | 1X4X8 | 2473.78 | 1X8X8 | 1379.25 | 44,25% |
| Pior 3 dimensões | 4X8X2 | 2611.71 | 8X2X2 | 1583.28 | 39,38% |
| Melhor 3 dimensões | 2X4X4 | 2527.33 | 8X2X4 | 1496.97 | 40,77% |

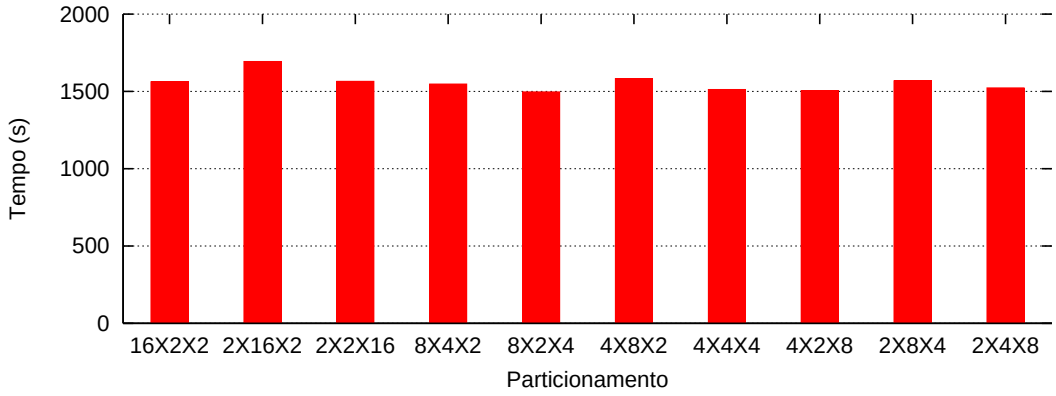


Figura 7.13: Tempo de execução obtido utilizando 64 processadores para diferentes configurações de particionamento em 3 dimensões do modelo tridimensional para um reticulado de 256 X 256 X 256 pontos

### 7.2.2 Análise dos Resultados Computacionais

As divisões por blocos e cubos apresentaram significativas diminuições do tempo total de execução do modelo 3D para o reticulado de 128 X 128 X 128 elementos. Conforme apresentado nas Tabela 7.2, Tabela 7.3 e Tabela 7.4, à medida que um número maior de processadores era utilizado, aumentava a diferença entre o ganho de desempenho usando o particionamento em blocos em relação ao particionamento linear. Com isso, nas simulações com 40 processadores o ganho de desempenho apresentado pelos melhores casos chegou a ser até 31% superior em relação ao particionamento linear.

Além de reduzir os tempos de execução, as implementações em bloco mostraram-se também mais escalares em relação ao particionamento linear. Comparando-se os valores apresentados na Figura 6.13 e Figura 7.9 pode-se observar que o ganho de desempenho, que era de apenas 18 utilizando 40 processadores no particionamento linear, passou a ser de 25 para o particionamento em blocos.

Em termos de estratégias de particionamento, um fator de forte impacto nos resultados é de que, da mesma forma que no particionamento linear, o tempo de execução é pior para os casos em que o particionamento do eixo x é maior. Conforme ressaltado anteriormente, nesse caso haverá um sobrecarga oriunda da sincronização do cálculo da velocidade média em paralelo. Por essa razão, particionamentos em 3 dimensões, com os pontos dos sub-reticulados dispostos de uma forma mais cúbica, nem sempre apresentam resultados melhores em relação ao particionamento em 2 dimensões.

Os resultados obtidos a partir do estudo de caso utilizando um reticulado de 256 X 256 X 256 elementos avaliaram o comportamento do método em problemas de grandes dimensões. Através das execuções foi possível observar que o uso de 64 processadores possibilitou uma redução dos tempos de execução em quase pela metade quando comparados com os testes utilizando 32 processadores, o que indica uma boa escalabilidade para a implementação, embora é preciso novamente observar que entre esses 64 processadores, haviam dois grupos com pequenas diferenças em termos de capacidade de processamento. Diferente dos resultados apresentados para o reticulado de 128 X 128 X 128 elementos, a redução dos tempos de execução do particionamento em duas e três dimensões para o problema em questão não foi tão elevada em relação ao particionamento unidimensional utilizando 32 processadores. No entanto, utilizando tanto 32 como 64 processadores pode-se observar que os menores tempos de execução ficaram com as configurações de

particionamento onde os dados estavam dispostos em uma geometria mais quadrática ou cúbica.

# 8 CONCLUSÃO

Esta dissertação tratou do desenvolvimento de implementações paralelas do Método de Lattice Boltzmann (MLB) utilizando estratégias de particionamento de dados eficientes para reticulados com estruturas de regulares. As implementações propostas englobam os principais modelos bidimensional e tridimensional do método encontrados na literatura. Tais modelos possibilitam a modelagem de diversos fenômenos físicos de grande importância para a pesquisa em Dinâmica de Fluidos Computacional (DFC). Os resultados obtidos através da simulação dos estudos de caso permitiram constatar que o particionamento em blocos pode oferecer tanto um significativo aumento de desempenho quanto uma maior escalabilidade para as implementações paralelas do método. Esses resultados coincidem com as expectativas oriundas das análises teóricas previamente feitas, comprovando a eficiência das técnicas de paralelização em blocos como forma de incrementar o desempenho do método em ambientes com memória distribuída.

## 8.1 Revisão do Trabalho Desenvolvido

Inicialmente, no Capítulo 1, foi feita a introdução do trabalho. Para tanto foram apresentados o contexto de desenvolvimento do MLB em relação à outras técnicas de representação e simulação computacional de fluxos de fluido, bem como as abordagens de paralelização geralmente aplicadas no cálculo numérico. Foram definidos ainda os objetivos e a contribuição da dissertação.

Os Capítulos 2 e 3 trataram do contexto científico do trabalho. No Capítulo 2 foram definidas as equações de Euler, de Navier-Stokes e de Boltzmann. Tais equações foram relacionadas às propriedades microscópicas dos fluidos através da Equação de Boltzmann. Através dessa equação, é possível afirmar que simulações microscópicas ou mesoscópicas tendem a gerar os mesmos resultados físicos que a resolução numérica das equações macroscópicas.

O Capitulo 3 apresentou o Método de Lattice Boltzmann, discutindo suas características. Na primeira parte do capítulo foi descrita a técnica de Lattice Gas Automata. Na evolução das técnicas de simulação mesoscópica, o MLB manteve o modelo reticular e as regras de propagação das partículas utilizadas por Lattice Gas Automata. Por outro lado, a utilização de valores reais, ao invés de lógicos, para as operações de propagação e colisão tornaram o método similar a equação discreta de Boltzmann. A partir do desenvolvimento das equações apresentadas para o método pode-se obter um algoritmo relativamente simples, composto essencialmente de operações de propagação, colisão, mecanismo de controle das condições de contorno e cálculo de valores macroscópicos. Apesar da simplicidade do algoritmo, o método pode ser utilizado em diferentes contextos, obtendo valores similares aos apresentados pelos modelos macroscópicos, conforme

a discussão sobre a relação do MLB com as Equações de Navier-Stokes.

Uma vez definido o contexto científico, fez-se a revisão bibliográfica do trabalho no Capítulo 4. Como estado da arte em termos de programação paralela foi mencionada a biblioteca de comunicação MPI. Os recursos de programação apresentados pela biblioteca abrangem muitos recursos de comunicação eficientes, sendo estes ideais para o desenvolvimento de aplicações paralelas com alto grau de desempenho. Particularmente, os recursos de comunicação cartesiana permitem implementar o particionamento dos dados em blocos. Apesar dessa técnica de particionamento já ter sido utilizada em implementações paralelas do MLB, conforme apresentado em algumas referências, nenhuma análise de desempenho ainda havia sido feita em relação as diferentes formas de particionar o reticulado.

No Capítulo 5 foram descritas as implementações paralelas do MLB, utilizando o particionamento de dados em bloco. Uma vez definido os detalhes técnicos, foram feitas avaliações teóricas sobre o modelo de comunicação adotado para cada uma das implementações realizadas. Para tantoi, adotou-se o modelo de máquina paralela LogP. Esse modelo leva em consideração os custos de comunicação, sendo por isso, ideal para sistemas de memória distribuída. Através dos exemplos desenvolvidos pode-se perceber que o particionamento em blocos consegue diminuir o custo de comunicação em mais da metade se comparado ao particionamento monodimensional.

Uma vez definidas as implementações, o Capítulo 6 apresentou o contexto das avaliações, a descrição dos estudos de caso, os resultados físicos e os resultados computacionais obtidos utilizando o particionamento linear. Os testes foram feitos utilizando 3 estudos de caso elaborados artificialmente. Os resultados físicos foram condizentes com as condições físicas pré-determinadas. Já os resultados computacionais obtidos através da simulação dos estudos de caso mostraram que a paralelização consegue diminuir o tempo total de execução do método. No entanto, o ganho de desempenho apresentou-se bastante limitado.

Como forma de superar o limite de desempenho do particionamento unidimensional, foram utilizadas estratégias de particionamento em blocos, cujos resultados foram mostrados no Capítulo 7. Além de comprovar as análises teóricas, as quais previam a diminuição do tempo de comunicação entre os processos, os resultados computacionais superaram os limites de ganho de desempenho que existiam no particionamento monodimensional. Dessa forma, o particionamento em blocos aumentou a escalabilidade das implementações paralelas do método.

## 8.2 Contribuições

O objetivo deste trabalho foi avaliar algumas estratégias de particionamento e distribuição de dados que tornam as execuções paralelas do MLB mais eficientes. A principal contribuição apresentada foi comprovar que técnicas de distribuição de dados não triviais, como é o caso do particionamento em blocos, quando testadas de forma prática em uma aplicação real, comprovam a eficiência apresentada nos modelos teóricos. Outras contribuições obtidas a partir de seu desenvolvimento desta dissertação foram:

- Disponibilizar implementações dos modelos de propagação de velocidade mais utilizados do MLB, tanto dos modelos bidimensionais como dos modelos tridimensionais;

- Paralelizar os modelos D2Q9 e D3Q19 do MLB através da técnica de particiona-

mento de dados, sendo a distribuição dos elementos do reticulado feita de forma balanceada;

- Refinar as implementações paralelas utilizando o particionamento dos dados em blocos;
- Realizar avaliações teóricas do custo de comunicação entre processos;
- Analisar os resultados obtidos utilizando o particionamento em blocos, cobrindo uma lacuna encontrada na literatura;
- Solucionar alguns estudos de caso relacionados ao escoamentos de fluxos de fluido através de canais com obstáculos utilizando as implementações do método;
- Disponibilizar ao grupo de pesquisa, uma nova técnica de simulação de fluidos, incorporada às soluções já utilizadas.

Durante o desenvolvimento do trabalho foram escritos alguns artigos e resumos, os quais foram submetidos em eventos regionais, nacionais e internacionais. Essas publicações relacionam algumas contribuições parciais obtidas em diversas etapas do andamento do trabalho. Os trabalhos podem ser encontrados no Anexo B. Para eventos internacionais da área foram submetidos artigos ao PARA06 (Umea, Suécia), SBAC 2006 (Ouro Preto, Brasil) e CCGrid 2007 (Rio de Janeiro, Brasil), tendo sido o primeiro aprovado. Em 2006 foi feita também uma submissão ao WPerformance, um evento nacional da área de avaliação de desempenho em sistemas computacionais. Além desses eventos, foram publicados ainda resumos nos eventos regionais ERAD 2006, Semana Acadêmica 2006/1, WSPPD 2006 e ERAD 2007.

## 8.3 Trabalhos Futuros

Os trabalhos futuros prevêem a continuidade do desenvolvimento, utilização e avaliação paralela do MLB. Nesse sentido, espera-se avaliar as implementações apresentadas neste trabalho em ambientes paralelos que disponham de um número maior de processadores, usando, para isso, tanto arquiteturas de *clusters* como *grids*. Conseqüentemente será possível aumentar o tamanho dos reticulados que serão utilizados como estudo de caso. Além dos ambientes de *cluster* e *grid*, arquiteturas dedicadas de alto desempenho, tais como arquiteturas com memória compartilhada, poderão ser adotadas, a fim de comparar o ganho de eficiência entre diferentes ambientes de execução.

Dentro dos trabalhos desenvolvidos pelo GPPD relacionados a Dinâmica de Fluidos, o estudo e utilização do MLB surge como uma técnica alternativa de simulação. Para o desenvolvimento de novas aplicações físicas pretende-se utilizar o método para agregar eficiência e precisão às soluções buscadas. Assim, pesquisas relacionadas com a modelagem de sistemas circulatórios sanguíneos, propagação dos ventos e simulação de rochas-reservatório poderão ser desenvolvidas de forma mais simples. Além disso, como trabalhos futuros é possível, ainda, avaliar o método realizando comparações com as demais técnicas já adotadas no passado pelo grupo.

Os protótipos e problemas desenvolvidos neste trabalho pretendem ser utilizados para validar o ambiente *Integrated Cluster Environment* (ICE), desenvolvido pelo GPPD. ICE é um ambiente integrado para a computação de alto desempenho via interface *web* (MARQUEZAN et al., 2006). Ele foi desenvolvido utilizando *Web Services* e possibilita o lançamento de aplicações paralelas em cluster através de uma interface específica e simples.

A idéia é usar as implementações do MLB feitas até o momento para avaliar criteriosamente a qualidade de ICE no que tange os parâmetros de entrada, lançamento, execução e obtenção de resultados pós processamento.

# REFERÊNCIAS

ANDERSON, J. D. **Computational Fluid Dynamics**. New York: McGraw-Hill, 1995. 547p.

ANDREWS, G. R. **Foundations of Multithreaded, Parallel, and Distributed Programming**. USA: Addison-Wesley, 2001.

ARTOLI, A.; HOEKSTRA, A.; SLOOT, P. Optimizing Lattice Boltzmann Simulations for Unsteady Flows. **Computers & Fluids**, [S.l.], v.35, n.2, p.227–240, 2005.

BARDOS, C.; UKAI, S. The Classical Incompressible Navier-Stokes Limit of the Boltzmann Equations. **Math. Models Meth. Appl. Sci.**, [S.l.], v.2, p.235–257, 1991.

BARRETT, R. et al. **Templates for the Solution of Linear Systems**: Building Blocks for Iterative Methods. Philadelphia, PA: SIAM, 1994.

BATCHELOR, G. K. **An Introduction to Fluid Dynamics**. [S.l.]: Cambridge University Press, 1987.

BHATNAGAR, P.; GROSS, E. P.; KROOK, M. K. A Model for Collision Processes in Gases. I. Small Amplitude Processes in Charged and Neutral One-Component Systems. **Phys. Rev.**, [S.l.], v.94, n.3, p.511 – 525, 1954.

BUICK, J. M. **Lattice Boltzmann Methods in Interfacial Wave Modelling**. 1997. PhD Dissertation — University of Edinburgh, Fluid Dynamics Group.

CARTER, J.; OLIKER, L. Performance Evaluation of Lattice-Boltzmann Magnetohydrodynamics Simulations on Modern Parallel Vector Systems. In: TERAFLOP WORKSHOP, 2., 2006, Stuttgart. **Proceedings...** [S.l.]: Springer, 2006.

CERCIGNANI, C. **The Boltzmann Equation and Its Applications**. Berlin: Springer, 1994. (Applied Mathematical Sciences, v.67).

CHEN, H.; CHEN, S.; MATTHAEUS, W. Recovery of Navier–Stokes equations using a lattice-gas Boltzmann method. **Phys. Rev. A**, [S.l.], v.45, p.R5339–42, 1992.

CHEN, S.; DOOLEN, G. D. Lattice Boltzmann Method for Fluid Flows. **Annual Review of Fluid Mechanics**, [S.l.], v.30, p.329–364, 1998.

CORREA, R. et al. **Models for Parallel and Distributed Computation**: Theory, Algorithmic Techniques and Applications. [S.l.]: Springer Verlag, 2002. 340p.

CULLER, D. E. et al. LogP: Towards a Realistic Model of Parallel Computation. In: ACM SIGPLAN SYMPOSIUM ON PRINCIPLES AND PRACTICE OF PARALLEL PROGRAMMING - PPOPP, 1993, New York, NY, USA. **Proceedings...** [S.l.]: ACM Press, 1993. p.1–12.

CULLER, D. E. et al. LogP: a practical model of parallel computation. **Communications of the ACM**, New York, NY, USA, v.39, n.11, p.78–85, 1996.

DESPLAT, J.-C.; PAGONABARRAGA, I.; BLADON, P. Ludwig: A parallel Lattice-Boltzmann code for complex fluids. **Computer Physics Communications**, [S.l.], v.134, n.3, p.273–290, 2001.

DONGARRA, J. The LINPACK Benchmark: an explanation. In: INTERNATIONAL CONFERENCE ON SUPERCOMPUTING, 1., 1988, London, UK. **Proceedings...** [S.l.]: Springer-Verlag, 1988. p.456–474.

DONGARRA, J. et al. (Ed.). **The Sourcebook of Parallel Computing**. [S.l.]: Elsevier, 2002.

DORNELES, R. V. **Particionamento de Domínio e Balanceamento de Carga no Modelo HIDRA**. 2003. Tese (Doutorado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre - RS.

DUPUIS, A. **From a lattice Boltzmann model to a parallel and reusable implementation of a virtual river**. 2002. PhD Dissertation — University of Geneva, CUI - Computer Science Departement - University of Geneva.

ELMROTH, E. et al. Recursive Blocked Algorithms and Hybrid Data Structures for Dense Matrix Library Software. **SIAM Rev.**, [S.l.], v.46, n.1, p.3–45, Mar. 2004.

Exa Corporation. **PowerFLOW for CFD - Driving Fluid Flow Simulation Technology Into the next Century**. Disponível em: <http://www.exa.com/newsite/frames/powerflowmaster.html>. Acesso em jun. 2006.

FANG, H. et al. Lattice Boltzmann method for simulating the viscous flow in large distensible blood vessels. **Physical Review E**, [S.l.], v.65, n.5, p.1–11, May 2002.

FERZIGER, J. H.; PERIC, M. **Computational Methods for Fluid Dynamics**. London, England: Springer Verlag, 2002.

FLEKKøY, E. G. Lattice Bhatnagar-Gross-Krook models for miscible fluids. **Physical Review E**, New York, USA, v.47, n.6, p.4247–4257, 1993.

FOSTER, I. **Designing and Building Parallel Programs**: Concepts and tools for Parallel Software Engineering. Reading, MA: Addison Wesley, 1995.

FRISCH, U. et al. Lattice Gas Hydrodynamics in two and Three Dimension. **Complex Systems**, [S.l.], v.1, p.649–707, 1987.

FRISCH, U.; HASSLACHER, B.; POMEAU, Y. Lattice-Gas Automata for the Navier-Stokes Equation. **Phys. Rev. Lett.**, [S.l.], v.56, p.1505, 1986.

GALANTE, G. **Métodos Multigrid Paralelos em Malhas Nao Estruturadas Aplicados à Simulação de Problemas de Dinâmica de Fluidos Computacional e Transferencia de Calor**. 2006. 130p. Dissertação (Mestrado em Ciência da Computação) — Instituto de Informática, UFRGS, Porto Alegre.

GEIST, A. et al. **PVM**: Parallel Virtual Machine: A Users' Guide and Tutorial for Networked Parallel Computing. Cambridge, MA, USA: MIT Press, 1994.

GIBBONS, P. B. A more practical PRAM model. In: ACM SYMPOSIUM ON PARALLEL ALGORITHMS AND ARCHITECTURES - SPAA, 1989, Santa Fe, New Mexico, United States. **Proceedings...** New York: ACM Press, 1989. p.158–168.

GROPP, W. et al. High-performance, portable implementation of the MPI Message Passing Interface Standard. **Parallel Computing**, [S.l.], v.22, n.6, p.789–828, 1996.

GUOI, Z.; ZHAO, T. S. Lattice Boltzmann simulation of natural convection with temperature-dependent viscosity in a porous cavity. **Progress in Computational Fluid Dynamics**, [S.l.], v.5, n.1/2, p.110 – 117, 2005.

HARDY, J.; POMEAU, Y.; PAZZIS, O. de. Time evolution of a two-dimensional model system. I. Invariant states and time correlation functions. **J. Math. Phys.**, [S.l.], v.14, p.1746, 1973.

HE, X.; LUO, L.-S. Theory of the lattice Boltzmann equation: from Boltzmann equation to lattice Boltzmann equation. **Phys Rev E**, [S.l.], v.56, p.6811–6817, 1997.

HE, X.; LUO, L.-S. Lattice Boltzmann model for the incompressible Navier-Stokes equation. **J. Stat. Phys.**, [S.l.], v.88, p.927–944, 1997.

HOCKNEY, R. W.; EASTWOOD, J. W. **Computer simulation using particles**. Bristol, PA, USA: Taylor & Francis, Inc., 1988.

JONSSON, I.; KÀGSTRÖM, B. Recursive blocked algorithms for solving triangular systems - Part I: one-sided and coupled Sylvester-type matrix equations. **ACM Trans. Math. Softw.**, New York, NY, USA, v.28, n.4, p.392–415, 2002.

KANDHAI, D. et al. Lattice-Boltzmann hydrodynamics on parallel systems. **Computer Physics Communications**, [S.l.], v.111, p.14–26, 1998.

KÖRNER, C. et al. Parallel Lattice Boltzmann Methods for CFD Applications. In: BRUASET, A.; TVEITO, A. (Ed.). **Numerical Solution of Partial Differential Equations on Parallel Computers**. [S.l.]: Springer Verlag, 2005. p.439–465. (Lecture Notes for Computational Science and Engineering, v.51).

KRAFCZYK, M. et al. Analysis of 3D transient blood flow passing through an artificial aortic valve by Lattice-Boltzmann methods. **Journal of Biomechanics**, [S.l.], v.31, n.5, p.453–462, May 1998.

LAM, M. D.; ROTHBERG, E. E.; WOLF, M. E. The cache performance and optimizations of blocked algorithms. In: INTERNATIONAL CONFERENCE ON ARCHITECTURAL SUPPORT FOR PROGRAMMING LANGUAGES AND OPERATING SYSTEMS - ASPLOS, 4., 1991, Santa Clara, California, United States. **Proceedings...** New York: ACM Press, 1991. p.63–74.

LAM/MPI Parallel Computing. Disponivel em: <http://www.lam-mpi.org>. Acesso em: out. 2006.

LANDAU, L. D.; LIFSHITZ, E. M. **Fluid Mechanics**. 2nd ed. Oxford, United Kingdom: Pergamon Press, 1982.

LOCKARD, D. P.; LUO, L.-S.; SINGER, B. A. **Evaluation of the Lattice-Boltzmann Equation Solver PowerFLOW for Aerodynamic Applications**. Disponível em: <http://www.engr.uky.edu/vac/public_html/CTEMPpowerflow.pdf>. Acesso em: out. 2006.

LUCQUIN, B.; PIRONNEAU, O. **Introduction to Scientific Computing**. New York, USA: J. Wiley & Sons, 1998. 380p.

MARQUEZAN, C.; RIGHI, R.; SCHNORR, L. M.; CARíSSIMI, A.; MAILLARD, N.; NAVAUX, P. O. A. ICE: A service oriented approach to uniform the access and management of cluster environments. In: INTERNATIONAL WORKSHOP ON GRID TESTBEDS; INTERNATIONAL SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID, 6., 2006, Washington, DC, USA. **Proceedings...** New York: IEEE Computer Society, 2006.

MASSELOT, A. **Lattice Gas Model for snow transport**. 2000. PhD Dissertation — University of Geneva, CUI - Computer Science Departement - University of Geneva.

MCNAMARA, G.; ZANETTI, G. Use of the Boltzmann Equation to Simulate Lattice-Gas Automata. **Phys. Rev. Lett.**, [S.l.], v.61, p.2332–2335, 1988.

MPI FORUM. MPI: A Message-Passing Interface Standard. **Journal of Supercomputing Applications**, [S.l.], v.8, n.3-4, p.165–414, 1994.

MPICH home page. Disponivel em: <http://www-unix.mcs.anl.gov/mpi/mpich1>. Acesso em: out. 2006.

MPICH2 Home Page. Disponivel em: <http://www.mcs.anl.gov/mpi/mpich>. Acesso em: out. 2006.

ONISHI, J.; CHEN, Y.; OHASHI, H. A Lattice Boltzmann model for polymeric liquids. **Progress in Computational Fluid Dynamics**, [S.l.], v.5, n.1/2, p.75 – 84, 2005.

OPEN MPI: Open Source High Performance Computing. Disponivel em: <http://www.open-mpi.org>. Acesso em: out. 2006.

PAN, C.; PRINS, J. F.; MILLER, C. T. A high-performance lattice Boltzmann implementation to model flow in porous media. **Computer Physics Communications**, [S.l.], v.158, n.2, p.89–105, 2004.

PHILIPPI, P. C. et al. Fluid interfaces in phase transition problems: Lattice-Boltzmann method. In: COBEM, 18., 2005, Ouro Preto - MG, Brazil. **Book of abstracts...** Rio de Janeiro: ABCM, 2005.

POHL, T. et al. Performance Evaluation of Parallel Large-Scale Lattice Boltzmann Applications on Three Supercomputing Architectures. In: ACM/IEEE CONFERENCE ON SUPERCOMPUTING - SC, 2004, Washington, DC, USA. **Proceedings...** [S.l.]: IEEE Computer Society, 2004. p.21.

QIAN, Y. H.; D'HUMIÈRES, D.; LALLEMAND, P. Lattice BGK Models for Navier-Stokes Equation. **Europhysics Letters**, [S.l.], v.17, p.479–484, Feb. 1992.

REIF, F. **Fundamentals of statistical and thermal physics**. [S.l.]: McGraw–Hill, 1965.

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

SANTOS, L. O. E. dos et al. Prediction of Intrinsic Permeabilities with Lattice Boltzmann Method. In: COBEM, 18., 2005, Ouro Preto - MG, Brazil. **Book of abstracts...** Rio de Janeiro: ABCM, 2005.

SIMS, J. S. et al. Accelerating Scientific Discovery Through Computation and Visualization. **Journal of Research of the National Institute of Standards and Technology**, [S.l.], v.105, n.6, p.875–894, Nov.-Dec. 2000.

SLOOT, P. et al. An Interactive Grid Environment for Non-Invasive Vascular Reconstruction. In: IEEE/ACM INTERNATIONAL SYMPOSIUM ON CLUSTER COMPUTING AND THE GRID - CCGRID, 4., 2004, Chicago, Illinois, USA. **Proceedings...** New York: IEEE, 2004. p.309–319.

SUCCI, S. **The Lattice Boltzmann Equation for Fluid Dynamics and Beyond**. New York, USA: Oxford University Press, 2001.

TECPLOT. Disponível em: <http://www.tecplot.com>. Acesso em: out. 2006.

VALIANT, L. G. A bridging model for parallel computation. **Communications of the ACM**, New York, NY, USA, v.33, n.8, p.103–111, 1990.

VERSTEEG, H.; MALALASEKRA, W. **An Introduction to Computational Fluid Dynamics**: The Finite Volume Method Approach. London, England: Springer Verlag, 1995.

WEI, X. et al. The Lattice Boltzmann Method for Gaseous Phenomena. **IEEE Transactions on Visualization and Computer Graphics**, [S.l.], v.10, n.2, p.164–176, Mar.-Apr. 2004.

WILKINSON, B.; ALLEN, M. **Parallel Programming**: Using Networked Workstations and Parallel Computers. USA: Prentice Hall, 1998.

WOLF, F. G.; SANTOS, L. O. E. dos; PHILIPPI, P. C. Formação e dinâmica da interface líquido-vapor simulada pelo método Lattice-Boltzmann. **Revista Brasileira de Ensino de Física**, [S.l.], v.28, n.2, p.167 – 175, 2006.

WOLF-GLADROW, D. A. **Lattice-Gas Cellular Automata and Lattice Boltzmann Models**: An Introduction. Berlin: Springer, 2000. 308p.

WU, H. R. et al. Lattice Boltzmann simulation of flow in porous media on non-uniform grids. **Progress in Computational Fluid Dynamics**, [S.l.], v.5, n.1/2, p.97 – 103, 2005.

XUAN, Y.; YU, K.; LI, Q. Investigation on flow and heat transfer of nanofluids by the thermal Lattice Boltzmann model. **Progress in Computational Fluid Dynamics**, [S.l.], v.5, n.1/2, p.13 – 19, 2005.

YU, D. et al. Viscous flow computations with the method of lattice boltzmann equation. **Progress in Aerospace Sciences**, [S.l.], v.39, n.5, p.329–367, July 2003.

# GLOSSÁRIO

**Autômato Celular** - São modelos matemáticos simplificados de iterações espaciais, em que cada ponto ou célula do espaço possui um estado próprio. A cada passo este estado é alterado segundo regras específicas condicionadas aos estados dos pontos vizinhos.

**Convecção** - Operação de deslocamento.

**Dinâmica de Fluidos Computacional (DFC)** - É a ciência que trata da obtenção de soluções numéricas das equações que governam um fluxo de fluidos através da simulação computacional. Desta forma, é possível calcular o valor das propriedades físicas através da simulação da variação de tempo e espaço.

**Densidade** - É a razão da massa pelo volume de uma determinada matéria.

**Difusão** - Expansão das moléculas (distribuição).

**Energia Interna** - É o total da energia de um sistema provocado pelo movimentação das partículas.

**Fluido** - Um fluido é um subconjunto de estados da matéria descritos como incapazes de resistir a uma deformação e capazes de escoar pelo espaço. Fluidos são representados por funções que descrevem suas propriedades físicas.

**Fluido Ideal** - É um fluido incompressível, com viscosidade nula, não turbulento . Muitos modelos baseiam-se em Fluidos Ideais devido a simplificação das operações.

**Fluido Real** - São os fluidos encontrados na natureza, geralmente difíceis de serem representados numericamente. Ao contrário dos Fluidos Ideais, os Fluidos Reais apresentam-se compressíveis e com viscosidade.

**Ganho de desempenho** - Utilizado como sinÔnimo para *speedup* ou fator de aceleração.

**Grade** - Uma malha disposta de forma ordenada.

**Gradiente** - Soma do vetor espacial de derivadas parciais para um espaço Euclidiano de $n$ dimensões.

**Hemodinâmica** - Estudo da circulação sanguínea e de seus componentes.

**Homogêneo** - Composto por uma só substância com as mesmas propriedades físicas.

**Imiscíveis** - Substâncias que não podem ser misturadas umas com as outras. Exemplo: água e óleo.

**Incompressível** - Um fluido é incompressível quando possui densidade constante, ou seja, apesar de todo o fluido poder ser compressível sob determinadas condições, ele não muda com o passar do tempo em um determinado domínio.

**Isotropia/Isotrópica** - Possuir as mesmas propriedades físicas em toda a composição do fluido.

**Irrotacional** - Um campo vetorial onde para um vetor $v$ tem-se $\nabla.v = 0$.

**Laminar** - Relativo a fluxos não turbulentos, ou seja, fluxos com baixo número de Reynolds.

**Laplaciano** - O operador de Laplace é um operador diferencial de segunda ordem em um espaço Euclidiano de $n$ dimensões definido como a divergência do gradiente. O Laplaciano é a soma de todas as derivadas parciais de segunda ordem.

**Magnetohidrodinâmica** - Estudo das interações entre eletromagnetismo e hidrodinâmica. As interações ocorrem geralmente em um fluido condutor de eletricidade, quando este se encontra envolvido por um campo magnético. Todo estudo teórico e fenomenológico da magnetohidrodinâmica pode ser feito através da utilização das equações de Maxwell, que governam o eletromagnetismo e as equações de Navier-Stokes. É conhecido também como magnetofluidodinâmica, magnetodinâmica ou hidromagnetismo.

**Miscíveis** - Propriedade química de duas ou mais substâncias que podem ser misturadas entre si (solúveis uma na outra). Exemplo: água e etanol.

**Modelo Molecular** - Simulação que considera cada uma das moléculas de um fluido.

**Momento** - A massa multiplicado pela velocidade.

**Número de *Knudsen*** - Número adimensional definido como a razão entre o comprimento médio do trajeto livre das moléculas de um fluido em relação a um comprimento característico. É utilizado para descrever o fluxo de um fluido de baixa densidade.

**Número de *Mach*** - É o valor obtido entre a razão da velocidade de um objeto ou fluido pela velocidade do som.

**Reticulado** - Também conhecido com *Lattice*. É uma estrutura geométrica de $n$ dimensões formada por diversos pontos dispostos de forma estruturada. Os pontos são organizados em um padrão regular periódico, tanto em duas como em três dimensões, sendo conectados por arestas.

**Macroscópico** - São as propriedades consideradas de um fluido em relação a sua composição elementar total, tais como temperatura, pressão e densidade.

**Mesoscópico** - São as propriedades intermediárias de um fluido. Neste caso, ao invés de considerar o fluido como um todo ou cada uma das moléculas individualmente, representa-se o mesmo como um conjunto de "pontos", de forma que cada um desses "pontos"seja composto por uma determinada quantidade de moléculas, o suficiente para determinar as propriedades físicas da substância.

**Microscópico** - Relativo as propriedades microscópicas, ou seja as propriedades e as interações moleculares do fluido.

**Momento** - Produto entre a massa e a velocidade.

**Número de Reynolds** - É a relação entre as propriedades de um fluido que definem características de comportamento do mesmo. É definido como $Re = \rho vd/h = vd/\tau$. Para Re > 30 um fluido se torna turbulento.

**Partícula** - É um conjunto de moléculas cujas propriedades são representadas por um único ponto.

**Subreticulado** - É parte de um reticulado. Ou seja, é uma das unidades que compõe o reticulado que foi particionado.

**Viscosidade** - Medida de resistência de um líquido em relação a um fluxo. É a quantidade de movimento que ocorre para o transporte microscópico por difusão molecular. Quanto maior a viscosidade, menor é a velocidade que um fluido se movimenta.

# ANEXO ARTIGOS

Neste anexo encontram-se relacionadas todas as publicações feitas durante o período de desenvolvimento desta dissertação.

## Publicações Aceitas

- "Uso do Método de Lattice Boltzmann em Aplicações da Hidrodinâmica". ERAD 2006, Ijuí - RS.
- "Parallel Implementation of the Lattice Boltzmann Method for Clusters". PARA'06, Umea, Suécia.
- "Implementação Paralela de uma Versão Bidimensional do Método de Lattice Boltzmann para Fluxos de Fluidos em Estruturas com Obstáculos". WSGPPD 2005, Porto Alegre - RS.
- "Paralelização do Modelo Bidimensional do Método de Lattice Boltzmann". ERAD 2007, Porto Alegre - RS.

## Publicações Submetidas

- "Uso de Aplicações Científicas para a Avaliação de Arquiteturas Paralelas". WPerformance 2006, Campo Grande - MS.
- "Simulating Fluid Flows in Complex Geometries Using a Parallel Lattice Boltzmann Implementation". SBAC-PAD 2006, Ouro Preto - MG.
- "Performance Improvement of the Parallel Lattice Boltzmann Method Through Blocked Data Distributions". CCGrid 2007, Rio de Janeiro - RJ (em avaliação).