Gerenciamento do Cálculo de Forças e Divisão de Trabalho em uma Implementação Paralela de Simulação de Dinâmica Molecular

Gonzalo Travieso
Jan Frans Willem Slaets

Grupo de Instrumentação Eletrônica e Informática
Departamento de Física e Ciência dos Materiais
Instituto de Física e Química de São Carlos
Universidade de São Paulo

Av. Dr. Carlos Botelho, 1465. CEP 13560-250
São Carlos - SP
e-mail: gonzalo@ifqsc.usp.br

Resumo

Descrevemos aqui uma reformulação do método de paralelização de dinâmica molecular proposto em [Tra92]. Mostramos que o novo método apresenta melhor desempenho, e apresentamos resultados de execução em uma rede de *transputers*.

Abstract

This work proposes some changes in the paralelization technique for molecular dynamics used on a previous paper [Tra92]. We show that the new method gives better performance, and present some execution results in a network of *transputers*.

1 Introdução

Em um artigo anterior [Tra92], apresentamos um método de paralelização de dinâmica molecular para sistemas que trabalham sob o paradigma de memória distribuída com passagem de mensagens, como é o caso de uma rede de *transputers*.

Naquele trabalho utilizamos, para a otimização do cálculo das forças entre pares, um algoritmo conhecido como *algoritmo de células* [AT87]. Este algoritmo foi utilizado pois apresenta as seguintes propriedades:

- Crescimento linear do tempo de processamento com o número de partículas
- Facilidade de paralelização através da divisão "geométrica" do trabalho entre os diversos processadores

Neste artigo propomos a utilização de um outro algoritmo, que consiste na combinação do algoritmo de células com outro algoritmo, conhecido como algoritmo de *tabela de vizinhos com atualização infreqüente*. Este último é um algoritmo inicialmente utilizado por Verlet [Ver67], e cuja combinação com o de células tem sido proposta na literatura (ver, por exemplo [Abr86]). Este algoritmo combinado apresenta ainda as vantagens apresentadas para o algoritmo de células, mas reduz sensivelmente o tempo de processamento, especialmente no caso de simulações tridimensionais [Tra93].

Propomos aqui também uma alteração, em relação a [Tra92], da forma como a divisão geométrica é realizada entre os processadores.

2 Descrição do método proposto

A fim de possibilitar uma análise do método proposto, apresentaremos inicialmente uma breve descrição do problema, apenas em seus aspectos significativos para o assunto em questão. Em seguida apresentaremos os algoritmos e a forma de divisão proposta.

2.1 Breve descrição do problema

O problema consiste em calcular a evolução temporal de um sistema de N partículas, sujeitas à interação mútua por um potencial especificado, através da integração das equações de movimento do sistema.

As interações envolvidas são de curto alcance, o que significa que, quando duas partículas estão a distância maior que uma distância especificada r_c, considera-se que não existe interação entre essas partículas. Ver figura 1. De forma a eliminar efeitos de borda (ver [AT87]), o espaço onde as partículas são distribuídas é considerado repetido periodicamente de forma infinita em todas as direções.

O intervalo de tempo Δt escolhido para a integração numérica deve ser suficientemente pequeno, de forma a que os erros introduzidos pela aproximação numérica sejam pequenos. Isso faz com que os deslocamentos sofridos pelas partículas dentro de um certo número de passos de integração seja pequeno, o que, como veremos a seguir, é importante para permitir uma otimização do cálculo de interações.

A parte do processamento que mais demanda recursos computacionais é o cálculo das forças de interação entre as diversas partículas. Em princípio, devemos calcular *todas* as distâncias entre partículas, de forma a permitir encontrar aquelas que distam menos

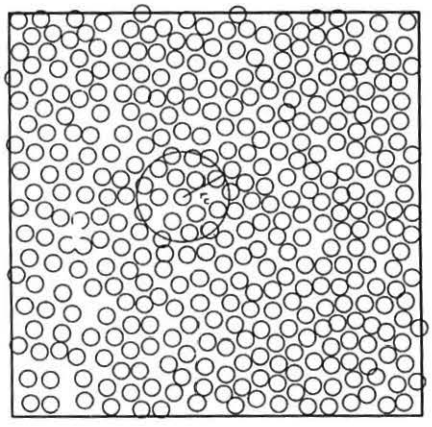

Figura 1: Partículas distribuídas no espaço de simulação

do que r_c, que seriam as que efetivamente interagem. No entanto, isso implicaria no cálculo de distâncias para $\mathcal{O}(N^2)$ pares de partículas o que, para grandes quantidades de partículas, dominaria todo o tempo de processamento. Veremos alguns algoritmos para lidar com esse problema.

2.2 Algoritmo de tabela de vizinhos

O algoritmo de tabela de vizinhos com atualização infreqüente, se baseia no fato de que as partículas do sistema se movem lentamente, no sentido de que dentro de um certo número de intervalos de integração as partículas permanecem aproximadamente nos mesmos locais, de forma que a lista de partículas que interagem com cada partícula dada muda muito pouco.

Utilizando este fato, o algoritmo consiste em montar, para cada partícula, uma tabela com todas as partículas que se encontram dentro de uma certa distância r_v maior que r_c. Desta forma, as interagentes com essa partícula serão encontradas entre as que est ao nessa tabela, e como as partículas se movem "lentamente", isto será válido para um certo número de passos de integração. A diferença entre os valores r_v e r_c, juntamente com o valor de Δt, determina o número de intervalos pelo qual é seguro se utilizar a mesma tabela. Assim, a atualização da tabela não precisa ser feita em todos os instantes, mas apenas a certos intervalos.

A vantagem desse algoritmo é que limita a fase que é proporcional a N^2 a apenas alguns intervalos de tempo. No entanto, continua existindo uma fase proporcional a N^2, que pode dominar o tempo de processamento quando o número de partículas for suficientemente grande.

2.3 Algoritmo de células

A implementação do algoritmo de células se baseia no fato de que as interações entre as partículas são de curto alcance. Assim, ao calcularmos as interações com uma dada

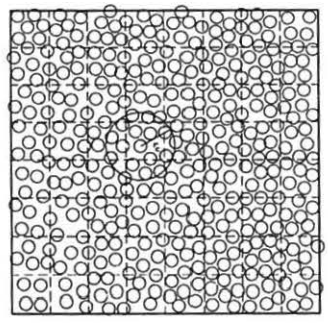

Figura 2: Divisão do espaço em células

célula, não necessitamos considerar todas as partículas, mas apenas as que se encontram espacialmente próximas da partícula dada. O modo de determinar quais são espacialmente próximas é através da divisão do espaço em um certo número de células, cada uma com um igual ou maior que r_c, conforme indicado na fig. 2. Assim, ao determinarmos as interagente com uma dada partícula, precisamos considerar apenas as partículas que se encontram na mesma célula ou em alguma das células vizinhas imediatas dessa célula.

Este método apresenta a vantagem de que elimina a fase proporcional a N^2, pois em nenhum instante necessitamos considerar todos os pares possíveis de partículas.

A desvantagem é que o número de partículas para as quais necessitamos calcular a distância com cada partícula é maior do que no caso do algoritmo de tabela de vizinhos com atualização infreqüente.

2.4 Algoritmo combinado

Podemos utilizar uma combinação dos dois algoritmos acima, da seguinte forma:

1. montamos uma estrutura de células com lado igual ou maior que r_v, ao invés de r_c;
2. à partir dessa estrutura de células, construímos a tabela de vizinhos;
3. realizamos os cálculos utilizando a tabela de vizinhos;
4. a atualização tanto da tabela de vizinhos como da estrutura de células é realizada infreqüentemente, a intervalos determinados pelos mesmos fatores que determinavam os intervalos no caso do algoritmo de tabela de vizinhos.

Este esquema combina as vantagens dos dois métodos anteriores: o número de partículas que precisam ter suas distâncias calculadas é determinado pelo tamanho da tabela de vizinhos (que é menor que a quantidade de partículas nas células vizinhas), e a montagem da tabela de vizinhos não apresenta uma fase proporcional a N^2, pois utiliza a estrutura de células.

Propomos portanto a utilização desse algoritmo combinado, em lugar do algoritmo de células proposto em [Tra92].

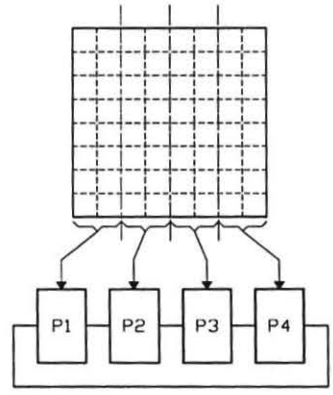

Figura 3: Divisão das células entre os processadores

2.5 Divisão homogênea × não-homogênea

A divisão do espaço em células permite uma forma de paralelização simples, que consiste em dividir parte das células para cada um dos processadores disponíveis. Devido a questões que não abordaremos aqui (ver [Tra93]), a forma natural de divisão é através de fatias numa mesma direção, conforme representado na figura 3.

No entanto, aqui surge um problema: como agir quando o número de células em cada direção não é múltiplo do número de processadores a ser utilizado?

Existem basicamente duas formas de realizar a divisão:

Divisão não-homogênea onde distribuímos as células conforme estão, em fatias verticais, entre os processadores, ficando então alguns processadores com carga maior que outros (fig. 4)

Divisão homogênea onde recalculamos o tamanho da célula, de forma a garantir que o número de células na direção da divisão seja múltiplo do número de processadores (fig. 5)

O primeiro método tem a desvantagem de introduzir um desbalanço de carga entre os processadores, enquanto que o segundo método tem a desvantagem de implicar num aumento do tamanho das células, com o correspondente aumento do número de pares que devem ser considerados ao buscar os pares interagentes.

No entanto, quando é utilizado o método combinado, a desvantagem da divisão homogênea fica bastante reduzida, pois somente se manifestará nos intervalos (pouco freqüentes), em que se fizer necessário o novo cálculo das tabelas de vizinhos.

Assim, propomos a utilização da divisão homogênea, ao contrário da divisão não-homogênea utilizada em [Tra92].

3 Resultados

A tabela 1 apresenta uma comparação dos desempenhos dos algoritmos de células e combinado, tanto para o caso bidimensional como para o caso tridimensional. Vemos que o

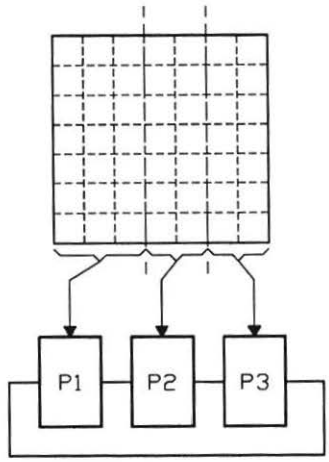

Figura 4: Divisão não-homogênea

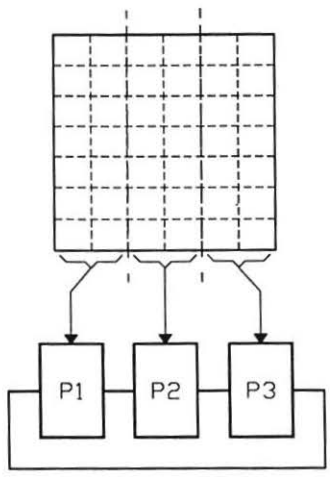

Figura 5: Divisão homogênea

Tabela 1: Comparação entre o método de células e o método combinado, com relação ao tempo de processamento, tanto para 2 como 3 dimensões. $N = 2048$, $\rho = 0.8442$.

Método	Tempo	
	2 dim.	3 dim.
Células	1.13	8.36
Combinado	0.77	3.56

Tabela 2: Tempos de simulação (em segundos) para o caso bidimensional

		Homogênea			Não-homogênea		
N	$P=1$	$P=2$	$P=3$	$P=4$	$P=2$	$P=3$	$P=4$
512	0.143	0.074	0.051	0.041	0.074	0.057	0.046
968	0.268	0.138	0.094	0.073	0.138	0.099	0.082
1458	0.404	0.209	0.140	0.108	0.211	0.145	0.118
2048	0.566	0.287	0.194	0.148	0.287	0.200	0.148
2450	0.677	0.342	0.231	0.178	0.342	0.250	0.192
2888	0.795	0.401	0.268	0.205	0.401	0.268	0.205

Tabela 3: Tempos de simulação (em segundos) para o caso tridimensional

		Homogênea			Não-homogênea		
N	$P=1$	$P=2$	$P=3$	$P=4$	$P=2$	$P=3$	$P=4$
864	1.041	0.537	0.384	0.290	0.538	0.513	0.291
1372	1.727	0.881	0.630	0.478	0.882	0.867	0.479
2048	2.508	1.309	0.917	0.681	1.507	1.022	1.025
2916	3.504	1.807	1.203	0.970	1.807	1.212	1.176
4000	4.930	2.496	1.684	1.376	2.499	1.696	1.683

método combinado apresenta vantagens significativas, especialmente no caso tridimensional.

Para comparar a divisão homogênea com a não-homogênea, veja-se as tabelas 2 e 3. Como vemos, a divisão homogênea apresenta-se em todos os casos vantajosa sobre a divisão não-homogênea (a menos dos casos onde o número de células já é múltiplo do número de processasores, caso em que as duas variantes são iguais).

Como complemento, apresentamos as figuras 6 e 7 que apresentam *speedups*, para até 4 processadores, tanto para o caso bidimensional como para o caso tridimensional, do sistema de processamento paralelo com a inclusão das alterações propostas. Vemos que o sistema apresenta um *speedup* próximo de linear dentro do número de processadores utilizados. Em [Tra93] apresentamos uma extrapolação desses resultados para maior número de processadores.

4 Conclusão

Com relação a um trabalho anteriormente apresentado, propusemos alterações que consistem em:

- Utilização de um algoritmo que combina os algoritmos de tabela de vizinhos com atualização infreqüente e o algoritmo de células;
- Introdução da divisão geométrica homogênea entre os processadores.

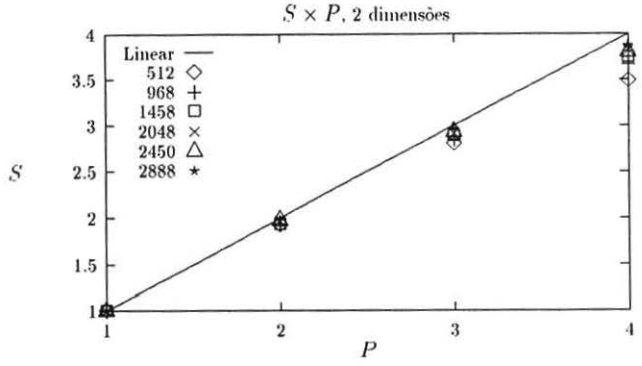

Figura 6: *Speedup* para o caso bidimensional

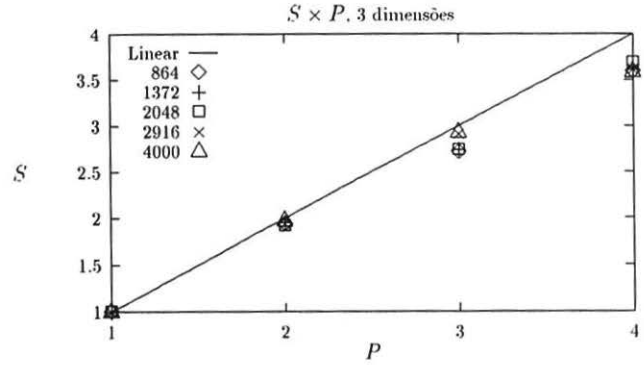

Figura 7: *Speedup* para o caso tridimensional

Mostramos que essas alterações introduzem uma melhor de desempenho do sistema paralelo, que apresentou *speedup* aproximadamente linear.

Referências

[Abr86] Farid. F. Abraham. Computation statistical mechanics — metodology, properties and supercomputing. *Adv. in Physics*, 35(1):1–111, 1986.

[AT87] M. P. Allen and D. J. Tildesley. *Computer simulation of liquids*. Claredon Press, 1987.

[Tra92] Gonzalo Travieso. Implementação de dinâmica molecular em uma rede de *transputers*. In *Anais do IV Simpósio Brasileiro de Arquitetura de Computadores - Processamento de Alto Desempenho*, p. 27–35, outubro 1992.

[Tra93] Gonzalo Travieso. *Proposta e implementação de um sistema de processamento paralelo para dinâmica molecular*. Tese de Doutorado, Instituto de Física e Química de São Carlos, maio 1993.

[Ver67] L. Verlet. Computer experiments on classical fluids: I. Thermodynamical properties of Lennard-Jones molecules. *Physical Review*, 159(1):98–103, july 1967.