PROPOSTA DE UMA ARQUITETURA DEDICADA À DINÂMICA MOLECULAR

G. Travieso , J. F. W. Slaets
Instituto de Física e Química de São Carlos
Universidade de São Paulo
Caixa Postal 369, 13560 - São Carlos, SP

RESUMO

Apresentaremos neste artigo um esboço de uma máquina dedicada à dinâmica molecular, baseada em uma rede de transputers T800. Numa fase inicial é feita uma definição breve do problema, seguida de uma análise seqüencial de eficiência de alguns algoritmos para o mesmo. A seguir, os dois algoritmos seqüenciais mais eficientes são estudados quanto à paralelização. Por fim, é apresentada a estrutura proposta para a rede de transputers.

ABSTRACT

This paper describes the study of a dedicated architecture, based on a T800 transputer network, to be used in molecular dynamics calculations. After a brief description of the physical problem, some performance analisys for sequential algorithms are presented. The parallel implementation of the two best optimized algorithms are then discussed. Finally, a parallel dedicated transputer based architecture is presented.

1. INTRODUÇÃO

O desenvolvimento de máquinas dedicadas a problemas específicos tem se constituido em uma prática cada vez mais comum quando se exige grande poder computacional em tarefas que apresentam a necessidade de repetição constante. As caracteristicas que levam à determinação da construção de uma máquina dedicada podem ser encontradas delineadas em |1|.
Um problema que apresenta a necessidade de construção de uma máquina dedicada é o de dinâmica molecular, como já foi apresentado em |2|.

2. DINÂMICA MOLECULAR

Apresentando-se de maneira simplificada, o problema de dinâmica molecular consiste no cálculo da evolução temporal das posições das partículas de um agregado, que interagem entre sí de acordo com uma função potencial definida. Em cada quadro da simulação são calculadas algumas variáveis do agregado, como por exemplo, energia cinética (que corresponderá à temperatura).

Alguns aspectos são importantes para a definição do problema com vistas ao desenvolvimento do sistema que realizará sua simulação:

a) escolhem-se condições de contorno periódicas, de forma que uma partícula, ao sair do espaço por uma de suas faces, imediatamente entra pela sua oposta;
b) somente necessitamos considerar, ao calcular a força sobre uma dada partícula, as partículas que se encontrarem a uma distância menor que o chamado "raio crítico".

Um requisito considerado fundamental para o sistema a ser desenvolvido é a expansibilidade do mesmo, de forma a permitir que o seu poder de computação cresça de acordo com a necessidade do usuário e a disponibilidade de verbas.

3. ALGORITMOS SEQÜENCIAIS

Passemos agora ao estudo de diversos algoritmos para a implementação seqüencial da simulação de dinâmica molecular.

3.1 Apresentação dos algoritmos.

3.1.1. Algoritmo básico.

Um algoritmo que implementasse diretamente o processo de cálculo envolvido na dinâmica molecular seria extremamente ineficiente, mesmo para quantidades pequenas de partículas. Por isto, não estudaremos aqui este algorítmo, que chamamos de algoritmo básico. O estudo do mesmo pode ser encontrado em |1|.

3.1.2. Algoritmo básico modificado.

Algumas pequenas modificações no algoritmo básico podem ser introduzidas de forma a conseguirmos uma redução bastante grande no tempo de execução. Essas alterações são:

a) considerar a simetria existente nas forças de interação entre as partículas (a força que a partícula i exerce sobre a partícula j é igual, porém de sinal contrário à que a partícula j exerce sobre a partícula i);
b) considerar a localidade da interação entre as partículas, realizando um teste da distância entre elas com o valor do raio crítico.

Com estas alterações simples, o novo algoritmo fica como descrito no pseudo-código abaixo:

```
para n de 1 até NQ faça:
| para i de 1 até N-1 faça:
| | para j de i até N faça:
| | | calcule a distância entre i e j
| | | se esta for menor que o raio crítico:
| | | | calcule força entre i e j
| | | | acumule força em acumuladores de i e j
| | | | calcule energia potencial entre i e j
| | | | acumule energia potencial
| para i de 1 até N faça:
| | calcule nova posição de i
| | calcule energia cinética de i
| | acumule energia cinética de i
| envie energias cinética e potencial
```

3.1.3. Algoritmo de tabela de vizinhos com atualização infreqüente.

Este algoritmo foi introduzido pela primeira vez por Verlet em 1967 |3|. O mesmo é sugerido pelo fato de que, nos algoritmos anteriores a maior parte do tempo é gasta nas fases de cálculo e teste de distância entre partículas. Isto se deve a ser esta fase proporcional ao quadrado do número de partículas, enquanto as outras fases são lineares com o mesmo.

Para reduzir o tempo gasto nesta fase, portanto, Verlet introduziu o expediente de realizá-lo apenas em alguns dos quadros de simulação. Isto pode ser feito sem comprometer os resultados, através do teste das distâncias não com o raio crítico em sí, mas com um "raio de vizinhança", suficientemente maior que o raio crítico de forma a garantir que nenhuma partícula exterior a esse raio de vizinhança cruze a película entre o raio de vizinhança e o raio crítico antes de um novo ciclo de teste de distâncias. Este algoritmo consiste portanto na formação de uma tabela de vizinhos, que possui, para cada partícula uma lista de todas as que com ela interagem, sendo que a atualização desta tabela é realizada em apenas alguns dos ciclos de simulação. Este algoritmo é apresentado no pseudo-código a seguir:

```
para n de 1 até NQ faça:
| se (n-1) MODULO txa = 0 :
| | para i de 1 até N-1 faça:
| | | para j de i até N faça:
| | | | calcule distância entre i e j
| | | | se distancia < raio de vizinhança:
| | | | | inclua j na tabela de i
| para i de 1 até N faça:
| | para j de 1 até NV|i| faça:
| | | calcule distância entre i e TV|i,j|
| | | calcule força entre partículas
| | | acumule forças totais de i e TV|i,j|
| | | com distância calcule energia potencial
| | | acumule energia potencial
| | calcule nova posição de i
| | calcule energia cinética de i
| | acumule energia cinética
| envie energias cinética e potencial
```

Neste pseudo-código incluímos algumas variáveis que são:

txa: taxa de atualização, representa o número de quadros de simulação que correrão sem atualização da tabela de vizinhos;
NV|i|: número de vizinhos da partícula i, encontrado durante a formação da tabela de vizinhos;
TV|i,j|: j-ésima vizinha da partícula i. A matriz TV é a própria tabela de vizinhos.

3.1.4. Algoritmo de granulação grosseira.

Este algoritmo representa uma reestruturação significativa do processo de cálculo da dinâmica molecular. Para compreender seu funcionamento, partamos da fig. 1. Nesta figura vemos o esquema de um espaço bidimensional no qual estão distribuídas as partículas. Este espaço foi dividido em partes menores, que chamaremos grãos. Notemos que a divisão foi realizada de forma a que o tamanho de cada grão corresponde, aproximadamente, ao raio crítico. Esta divisão nos permite garantir que, se uma partícula se encontra dentro de um dado grão, a mesma interagirá apenas com partículas presentes em um dos oito vizinhos desse grão (26 vizinhos para o caso tridimensional).

O algoritmo que denominamos de granulação grosseira utiliza este fato para conseguir linearizar o tempo de execução com o número de partículas do agregado |1|. Este algoritmo é apresentado no seguinte pseudo-código:

```
para n de 1 até NQ faça:
| para g de 1 até NG faça:
| | para i de 1 até NP|g| faça:
| | | para v de 1 até NV faça:
| | | | para j de 1 até NP|v| faça:
| | | | | calcule dist. entre P|g,i| e P|v,j|
| | | | | calcule força de interação
| | | | | acumule força em força sobre P|g,i|
| | | | | calcule energia potencial
| | | | | acumule energia potencial
| | | calcule nova posição de P|g,i|
| | | calcule novo grão de P|g,i|
| | | calcule energia cinética de P|g,i|
| | | acumule energia cinética
| envie energias cinética e potencial
```

Os novos fatores incluidos são:

NG: número total de grãos no espaço;
NP|g|: número total de partículas dentro do grão g;
NV: número de vizinhos por grão, incluindo-se o próprio grão entre eles;

P|g,i|: partícula i-ésima do grão g.

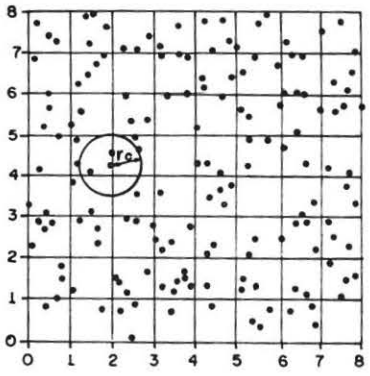

fig. 1: Divisão do espaço em grãos

3.2. Eficiência comparativa dos algoritmos apresentados.

Para um estudo da eficiência comparativa dos algoritmos apresentados anteriormente, nos basearemos em tempos de execução apresentados para o transputer T800 da INMOS em |4| e |5|, sendo que algumas simplificações, que não comprometem o objetivo de comparar os algoritmos, serão realizadas.

As simplificações principais envolvidas são:

a) consideramos que todo o programa, bem como todos os dados estão presentes na memória interna do T800.
b) os tempos de teste e atualização de variáveis dos diversos ciclos, por se tratarem de operações com números inteiros, são considerados desprezíveis;
c) desprezaremos todos os tempos gastos com inicialização de variáveis;
d) desprezaremos todos os tempos de controle de fluxo do programa, bem como de manipulação de processos.

Como veremos a seguir, as diferenças entre os diversos algoritmos são suficientemente significativas para permitir que sejam desprezadas as possíveis flutuações devidas a esses fatores.

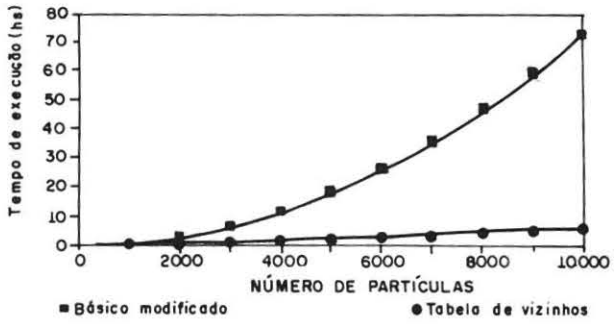

fig 2: Comparação dos algoritmos básico modificado e de tabela de vizinhos

As análises matemáticas de tempos de execução dos algoritmos, apresentadas em mais detalhes em |1|, nos levaram ao levantamento dos gráficos das figuras 2 e 3, onde são apresentados os valores de tempo para 1000 quadros de simulação, de forma comparativa, sendo que na fig. 2 comparamos os algoritmos basico modificado e de tabela de vizinhos e na fig. 3 os de tabela de vizinhos e granulação grosseira. A simples leitura desses gráficos nos permite salientar imediatamente os seguintes fatos:

a) a grande superioridade dos algoritmos de tabela de vizinhos e de granulação grosseira sobre o básico modificado;
b) a superioridade crescente do algoritmo de granulação grosseira sobre o de tabela de vizinhos quando aumentamos o número de partículas.

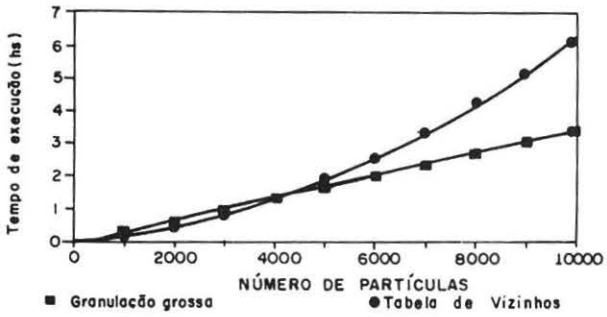

fig. 3: Comparação dos algoritmos de tabela de vizinhos e granulação grosseira

4. ESTUDO DE PARALELISMOS

Até agora estudamos o problema de dinâmica molecular de um ponto de vista exclusivamente seqüencial. Passemos agora a estudar os paralelismos, tanto do problema em sí como possíveis nos dois algoritmos mais eficientes apresentados.

4.1. Paralelismos inerentes ao problema.

Podemos notar que o próprio problema de dinâmica molecular apresenta certos paralelismos, que podem vir a ser explorados. Listemos a seguir os mesmos:

a) A interação entre as partículas é um processo simultâneo, o que permite a introdução de paralelismos no cálculo das novas posições das mesmas;
b) as forças e potenciais entre duas partículas quaisquer podem ser calculados sem levar em consideração qualquer fator que não as constantes do problema e as posições das duas partículas selecionadas, o que permite a realização em paralelo desses cálculos para quaisquer pares de partículas;
c) após calculada a distância entre duas partículas, podem ser efetuados em paralelo os cálculos de energia potencial e força entre partículas;
d) assim que esteja disponível a força que age sobre determinada partícula, é possível realizar-se o cálculo da sua nova posição;
e) após calculada a nova posição, o cálculo da energia cinética da partícula pode ser realizado;
f) os cálculos de distância, força, potencial, nova posição e energia cinética apresentam algumas partes independentes (e portanto paralelizáveis) para cada uma das dimensões do problema.

4.2. Paralelismo e o algoritmo de tabela de vizinhos.

O algoritmo de tabela de vizinhos constitui-se de duas fases, que correspondem à fase de cálculo da nova posição das partículas e à fase de atualização da tabela de vizinhos. Estas duas fases apresentam características distintas de paralelização e, portanto, serão estudadas separadamente.

Começando pela fase de cálculo da nova posição temos pronta, neste ponto, a tabela de vizinhos de cada uma das partículas do sistema, com as seguintes implicações com relação a paralelismo:

a) Se a tabela de vizinhos for apresentada conforme mostrado em 3.1.3., isto é, como uma relação das partículas que estão a uma distância menor que o raio de vizinhança da partícula considerada, surgiria um grave problema de concorrência no acesso de dados numa implementação paralela. Isto se deve a que, como não podemos estabelecer a priori quais as partículas que interagem entre sí, então é necessário que os dados sobre as posições de todas as partículas estejam disponíveis a todos os processadores que realizam os cálculos das novas posições das mesmas. Notemos que para complicar ainda mais a situação, esta concorrência cresceria proporcionalmente ao número de partículas do sistema. Esta dificuldade pode ser ultrapassada se constarem da tabela de vizinhos de cada partícula, não os identificadores de suas partículas vizinhas, mas sim as coordenadas das mesmas. Entretanto, a posição de todas as partículas se altera após os cálculos de um quadro de simulação, o que faria com que se tornasse necessário atualizar as novas coordenadas de todas as partículas em todas as tabelas onde elas aparecem, envolvendo uma quantidade muito elevada de comunicação entre processadores;
b) os cálculos de interação entre a partícula considerada e cada uma de suas vizinhas pode ser realizado independentemente, sendo que neste caso ocorre concorrência pelo acesso da tabela de vizinhos da partícula. Uma boa estratégia para lidar com esta concorrência é o processamento entrelaçado [6];
c) os cálculos de energia potencial e energia cinética totais exigem a comunicação dos dados parciais dos diversos processadores.

Na fase de atualização da tabela de vizinhos, podemos assinalar como mais importantes, com relação a paralelização, os seguintes pontos:

a) Para formar a tabela de vizinhos de cada partícula, o processador responsável pela mesma deve ter acesso às posições de todas as outras partículas do sistema, o que

implica numa grande concorrência de dados em um sistema paralelo. Novamente, uma excelente forma de lidar com esta concorrência é o processamento entrelaçado;

b) o cálculo da tabela de vizinhos é realizado apenas em alguns dos quadros da simulação, portanto é ineficiente deixar-se alocados para esse cálculo processadores específicos, que estariam ativos apenas raramente. Por outro lado, a utilização para estes cálculos dos mesmos processadores que realizam a atualização das posições das partículas apresenta o inconveniente de que as interligações entre os processadores têm que ser previstas para os dois processamentos diferente e, portanto, não podem ser otimizadas para nenhum deles.

Vemos portanto que este algoritmo apresenta sérios problemas para a sua implementação em um sistema paralelo.

4.3. Paralelismo e o algoritmo de granulação grosseira.

Veremos agora que este algoritmo apresenta melhores características de paralelização do que o de tabela de vizinhos com atualização infreqüente. Vejamos as suas características de paralelização:

a) Uma partícula de um dado grão necessita conhecer as coordenadas apenas das partículas vizinhas do mesmo e de seus vizinhos. Se os vizinhos de um grão estão localizados no mesmo processador, então não há nenhuma concorrência por dados. Se alguns dos grãos vizinhos de um dado grão está localizado em um processador diferente, então o processador do grão precisa se comunicar apenas com o processador desse grão vizinho (e não com qualquer processador arbitrário), portanto é necessária a comunicação apenas entre processadores vizinhos;

b) Depois de providenciado para que todos os dados de todos os grãos vizinhos a um grão específico estejam presentes, o cálculo da nova posição de todas as partículas desse grão pode ser realizado, inclusive com todas as possibilidades de paralelização apresentadas no ítem 4.1., pelo processador desse grão, independentemente de todos os outros;

Devido às vantagens deste algoritmo, o mesmo foi escolhido para a implementação da máquina de dinâmica molecular, cuja estrutura passaremos a descrever.

5. ESTRUTURA GERAL DA MÁQUINA PROPOSTA

No desenvolvimento de processadores dedicados a um dado problema, o ponto fundamental, que deve ser considerado durante todo o andamento do projeto, é o de alocar os recursos da máquina da forma mais eficiente possível ao problema em questão.

No nosso caso, escolhemos para a implementação física da máquina uma rede de transputers T800. A escolha destes se baseou principalmente nos seguintes fatos:

a) flexibilidade apresentada por uma rede de transputers, que pode ser facilmente alterada de forma a se adaptar a pequenas alterações do problema. Este fator não é fundamental numa máquina dedicada, mas se torna importante se considerarmos a necessidade de diluição dos custos da mesma por um certo número de pesquisadores;

b) representam uma alternativa de processamento paralelo de relativamente facil implementação, devido à sua própria estrutura, a linguagem (OCCAM) que o acompanha, e as ferramentas de desenvolvimento fornecidas pelo fabricante, o que permite concentrar a maior parte do esfoço de projeto nas fases de processamento paralelo propriamente dito.

A partir desta escolha, a tarefa de projeto se fundamenta na estruturação das interligações entre os processadores e na programação dos mesmos de modo que a execução seja o mais eficiente possível. Para se atingir este objetivo, devemos realizar o balanceamento de cargas entre os diversos elementos de processamento. As cargas de dois processadores são ditas balanceadas em um sistema de processamento paralelo quando as tarefas alocadas a estes dois processadores se cumprem no mesmo período de tempo. Isto corresponde à situação em que se verificam as duas seguintes condições:

1) nenhum processador suspende seu processamento por necessitar de dados de um outro processador;
2) não existem tempos inativos entre o término de uma tarefa e o início de outra para qualquer processador.

É intuitivo que quando as duas condições acima são simultaneamente satisfeitas tem-se eficiência máxima na utilização de todos os processadores do sistema. Note-se que o balanceamento perfeito de cargas é uma situação ideal que raramente é possível na prática, principalmente quando consideramos processamentos que possuem tempos de execução dependentes dos dados, como é o caso da dinâmica molecular.

Para tentarmos atingir um bom balanceamento de cargas e ao mesmo tempo providenciar a expansibilidade requerida para o sistema, propomos a sua execução em uma rede em forma de anel, sendo que cada nó do anel (que chamaremos de **divisão de processamento**), realizará o cálculo completo de dinâmica molecular sobre uma parte das partículas do agregado. A cada divisão de processamento serão associadas as partículas que se encontrem dentro de uma certa faixa do espaço de simulação, de forma que no seu conjunto as divisões de processamento cubram todo este espaço (e, portanto, todas as partículas).

Neste esquema, um excelente balanceamento de carga entre as diversas divisões de processamento pode ser conseguido facilmente simplesmente distribuindo a todas elas igual número de partículas.

A comunicação entre as divisões de processamento vizinhas precisa existir por dois fatores: em primeiro lugar, as partículas estão em movimento, o que faz com que elas continuamente troquem de divisão; em segundo lugar, para o cálculo das novas posições das partículas próximas ao limite de uma divisão necessitam dados sobre as partículas da divisão vizinha.

Se numerarmos os nós do anel seqüencialmente, de forma que os nós 1 e 2 sejão vizinhos, notamos que para calcularmos as novas posições das partículas que se encontram no grão mais à direita da divisão 1 devemos conhecer as posições das partículas do grão mais à esquerda da divisão 2 (pois estes grãos são vizinhos). Para evitar que a divisão 1 tenha que recorrer a dados existentes apenas na divisão 2 durante o processamento das novas posições (o que acarretaria a necessidade de constantes interrupções dos cálculos em uma divisão para a comunicação de dados, bem como constantes paradas na execução à espera de dados de divisões vizinhas), realizamos uma duplicação dos grão à esquerda da divisão 2 na 1, bem como dos mais à direita da divisão 1 na 2 (isto se repete para todas as divisões vizinhas). Com este expediente fazemos com que os dados precisem ser comunicados entre divisões vizinhas apenas após os cálculos das posições, o que impede que uma divisão necessite parar seus cálculos para esperar a comunicação de dados de uma vizinha.

Notemos que o processamento entre as diferentes divisões deve apresentar uma sincronização que impeça que, para o caso de existir diferença entre os tempos de simulação de um quadro entre as divisões, as mesmas não entrem em um estado em que as comunicações de dados entre elas percam o sentido por se referir a quadros distintos. Uma forma de sincronização que permite que algumas divisões se apresentem em um novo quadro enquanto outras se mantém no antigo pode ser realizado, e será apresentado em detalhes em |1|.

Para expandirmos o sistema, basta, na estrutura proposta acima, a inclusão de novos nós no anel, o que pode operar de duas formas distintas para o usuário: permitir a simulação com um número maior de partículas sem aumento de tempo de simulação ou permitir uma diminuição do tempo de simulação para um dado número de partículas.

Outro fator que também implica uma sincronização entre as diversas divisões de processamento é a necessidade de se calcular as energias cinética e potencial totais do agregado, o que implica a necessidade de dados de todas as divisões.

Ao lado desta divisão geométrica da simulação em divisões de processamento interligadas em anel, realizaremos uma divisão algoritmica dentro de cada uma das divisões de processamento. Isto é o que passaremos a tratar a seguir.

6. ESTRUTURA DAS DIVISÕES DE PROCESSAMENTO

Por ser um algoritmo suficientemente complexo, é útil que a simulação de dinâmica molecular com granulação grosseira seja dividida algoritmicamente (isto é, cada processador se incumbindo de uma fase do programa) entre um conjunto de processadores convenientemente interligados.

É claro que para este caso, o processo de balanceamento de cargas se torna mais complexo pois devemos seccionar o algoritmo e distribuir suas partes por processadores de tal forma que seja satisfeito o balanceamento de carga dos **processadores** e das **linhas de comunicação** entre os mesmos.

Considerando os cálculos envolvidos no algoritmo de granulação grosseira em conjunto com uma estimativa dos tempos de execução dos mesmos em um transputer T800, realizada a partir de dados de |4| e |5|, bem como estimativas médias do número de execuções de cada um dos cálculos, pudemos chegar a uma distribuição de cálculos pelos processadores e interligação entre os mesmos, como apresentada na fig. 4. Esta figura também indica os dados que circulam pelos "links" entre os diversos transputers.

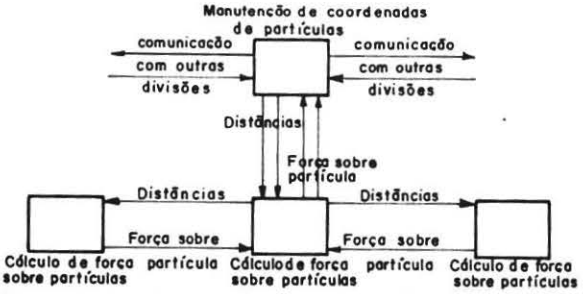

fig. 4: Estrutura das divisões de processamento

7. PROGRAMAÇÃO

A programação da máquina proposta será realizada diretamente no conjuto de instruções do T800, para permitir uma melhor otimização dos tempos de execução, bem como uma melhor utilização dos recursos do mesmo. Esta programação será formada de um conjunto de processos, distribuidos pelos transputers da fig. 4, de acordo com as considerações de balanço de carga dos processadores e das linhas de comunicação entre os mesmos, conforme definido acima.

8. CONCLUSÃO

Apresentamos um projeto para uma máquina dedicada a um problema simplificado de dinâmica molecular, delineando os passos fundamentais envolvidos em um trabalho deste tipo. Devemos acrescentar que o desenvolvimento apresentado aqui corresponde a uma simplificação de um trabalho atualmente em andamento e que será apresentado com detalhes em |1|.

REFERÊNCIAS

|1| **Travieso, G.**; dissertação de mestrado a ser submetida ao Instituto de Física e Química de São Carlos, para obtenção do grau de mestre em física aplicada

|2| **Travieso, G. , Slaets, J.F.W.**; "Máquina dedicada à dinâmica molecular", anais do I Simpósio Brasileiro de Arquitetura de Computadores - Processamento Paralelo, Gramado, maio de 1987

|3| **Verlet, L.**; "Computer 'experiments' on classical fluids. I. Thermodynamical properties of Lennard-Jones Molecules", Physical Review, vol 159, No. 1, julho de 1967

|4| ----------; "The transputer instruction set - a compiler writers' guide", INMOS, fevereiro de 1987

|5| **Askew, C.**; "T800 evaluation results", ESPRIT WP7: internal note #5, maio de 1987

|6| **Costa, L. da F., Travieso, G., Slaets, J.F.W.**; "Análise de eficiência e implementação de arquiteturas paralelas", anais do I Simpósio Brasileiro de Arquiteturas de Computadores - Processamento Paralelo