Simulação Computacional da Interação de Spins em Sistemas Magnéticos Usando Múltiplos Fluxos de Execução

João Paulo Peçanha- joaopaulo@ice.ufjf.br
Alessandra Matos Campos- amcampos@ice.ufjf.br
Rafael Barra de Almeida- faelrba@gmail.com
Patrícia Cordeiro Pereira Pampanelli- patricia.pampanelli@ice.ufjf.br
Marcelo Lobosco- marcelo.lobosco@ice.ufjf.br
Marcelo Bernardes Vieira- marcelo.bernardes@ufjf.edu.br
Departamento de Ciência da Computação
Sócrates de Oliveira Dantas- dantas@fisica.ufjf.br
Departamento de Física
Universidade Federal de Juiz de Fora, Cidade Universitária,
CEP: 36036-330 - Juiz de Fora, MG, Brasil

Resumo. *O estudo da interação de spins em compostos e elementos magnéticos vem se mostrando uma importante área de pesquisa por permitir uma melhor compreensão das propriedades magnéticas da matéria. Uma das abordagens utilizadas neste estudo é o emprego de modelos físicos, que então são simulados computacionalmente. Contudo, existe uma natural complexidade na resolução numérica dos modelos físicos, relacionada principalmente com o número de elementos presentes na estrutura simulada. Esta complexidade nos leva a buscar soluções computacionais que auxiliem na redução do tempo de processamento gasto na simulação. Assim sendo, este trabalho apresenta um modelo físico que descreve o comportamento de uma classe especial de materiais magnéticos, os ferromagnéticos, e um modelo computacional paralelo, baseado em múltiplos fluxos de execução e em memória compartilhada. Esta paralelização se mostra capaz de reduzir os custos associados aos cálculos relacionados a resolução numérica do modelo apresentado. Resultados preliminares indicam uma redução de até 3,85 vezes no tempo de execução, quando comparamos a versão seqüencial do simulador com a sua versão paralela.*

Keywords: *simulação física, interação de spins, simulação de materiais ferromagnéticos, paralelização de modelos físicos*

1. Introdução

A compreensão das origens das propriedades magnéticas dos materiais é de extrema importância para o desenvolvimento de novas tecnologias, com aplicações em distintas áreas como saúde, transportes, telecomunicações e armazenamento de dados. Para um melhor entendimento destas fenômenos, é fundamental o seu estudo em escala nanométrica, visto que a origem do magnetismo está associada a duas importantes propriedades dos elétrons nesta escala: a) o momento angular, associado ao movimento dos elétrons ao redor do núcleo atômico e b) o momento de *spins*, que está relacionado com a forma no qual os elétrons se distribuem na orbital do átomo.

A modelagem computacional destaca-se como uma importante ferramenta para o estudo

dos fenômenos magnéticos. O desenvolvimento de simuladores têm a grande vantagem de permitir a análise de distintos modelos, cada um empregando diferentes configurações, com menor custo financeiro e, em alguns casos, em menos tempo. Em trabalhos anteriores (Peçanha et al., 2008) apresentamos o MCSE (*Monte Carlo Spin Engine*), um simulador de spins em estruturas tridimensionais genéricas formadas por átomos com propriedades magnéticas. Entretanto, verificamos que a simulação dos modelos envolve um alto custo computacional, diretamente associado à quantidade de elementos presentes no sistema simulado, o que, em última instância, limita o tamanho do sistema que pode ser simulado.

Neste trabalho buscamos alternativas que tornem computacionalmente viáveis simulações com uma grande quantidade de spins, necessárias para reduzir os efeitos impostos pela obrigatoriedade de se tratar o problema de forma discreta. Em particular, neste trabalho empregamos processamento paralelo em memória compartilhada, dividindo as tarefas computacionais a serem executadas entre múltiplas linhas de execução (*threads*). A distribuição de tarefas é feita dinamicamente através do particionamento espacial da estrutura sendo simulada. Resultados preliminares indicam que esta abordagem permite um ganho de desempenho de até 3,85 vezes, quando comparamos os tempos da versão paralela de nosso simulador com sua versão seqüencial.

Este trabalho está assim organizado: as seções 2. e 3. apresentam, respectivamente, os modelos físico e computacional na qual se baseiam nosso simulador. O esquema empregado para a paralelização é também apresentado na seção 3., enquanto os resultados obtidos pela implementação deste esquema são apresentados na seção 4.. A seção 5. apresenta os trabalhos correlatos, enquanto a seção 6. conclui o trabalho.

2. Modelo Físico

Toda matéria apresenta propriedades magnéticas quando sob a ação de um campo magnético. Entretanto cada material reage ao campo magnético de uma forma, de acordo com as características físicas de seus átomos e das interações entre estes, o que permite classificá-los conforme a sua *susceptibilidade* $\chi = M/B$ (onde B é o campo externo e M a magnetização) em: ferromagnético e ferrimagnética ($\chi \gg 1$); diamagnética ($\chi < 1$); paramagnética ($\chi > 0$); antiferromagnética (χ pequeno) e materiais supercondutores ($\chi = -1$)(dos Santos Cabral Neto, 2004). Em especial, neste trabalho será analisado o comportamento dos materiais ferromagnéticos.

A magnetização dos materiais paramagnéticos e ferromagnéticos pode ser associada aos momentos magnéticos permanentes de seus átomos. De uma forma geral, o momento magnético de um átomo está relacionado com seu momento angular, conhecido como *spin*. Em mecânica quântica o *spin* de um átomo refere-se às possíveis orientações que partículas subatômicas têm quando estão sob ação, ou não, de um campo magnético externo.

Os materiais paramagnéticos são os que apresentam *suscetibilidade* magnética positiva muito pequena. Nestes materiais a interação entre *spins* é considerada fraca e, na ausência de um campo externo ($\vec{B}$), seus *spins* adotam orientações aleatórias. Devido a sua baixa *suscetibilidade*, os *spins* dos materiais paramagnéticos tendem, quando sob a ação de um campo externo, a se alinhar fracamente a este.

Os materiais ferromagnéticos apresentam características comportamentais opostas aos paramagnéticos. Seus átomos exercem forte influência nos *spins* dos átomos vizinhos, o que promove a criação de pequenos domínios magnéticos ao longo do material. Dentro desses domínios, os *spins* dos átomos adotam uma mesma orientação. Os ferromagnetos apresentam uma alta *suscetibilidade* a um campo magnético externo, portanto, um pequeno campo externo pode provocar um alto grau de alinhamento de seus *spins*.

Para simular computacionalmente o comportamento dos materiais ferromagnéticos é adotado o modelo de Heisenberg. Para esse modelo, o *spin* de um átomo i pode assumir qualquer direção ao longo de $\mathbb{R}^3$, portanto, $\vec{S}_i = (S_i^x, S_i^y, S_i^z)$ com $\left|\vec{S}_i\right| = 1$. O modelo de Heisenberg foi apresentado em 1928 e os detalhes do modelo podem ser consultados em (Landau & Binder, 2005).

Para n átomos situados em uma grade de simulação regular e igualmente espaçados, o Hamiltoniano que descreve a interação dipolar, ferromagnética e a interação com o campo magnético externo entre seus *spins* é dado por:

$$\mathcal{H} = \frac{A}{2} \sum_{\substack{i,j=1 \\ i \neq j}}^{n} \omega_{ij} - J \sum_{\substack{i,k=1 \\ i \neq k}}^{n} \vec{S}_i \cdot \vec{S}_k - \sum_{i=1}^{n} D(\vec{S}_i \cdot \vec{B}) \tag{1}$$

e ω_{ij} é definido por:

$$\omega_{ij} = \frac{\vec{S}_i \cdot \vec{S}_j}{\left|\vec{r}_{ij}\right|^3} - 3\frac{\left(\vec{S}_i \cdot \vec{r}_{ij}\right)\left(\vec{S}_j \cdot \vec{r}_{ij}\right)}{\left|\vec{r}_{ij}\right|^5} \tag{2}$$

onde $\vec{S}_i$ é o clássico *spin* de Heisenberg na posição $\vec{r}_i = (x_i, y_i, z_i)$ e $\vec{r}_{ij} = \vec{r}_i - \vec{r}_j$ é o vetor posição que separa os átomos i e j. As constantes A, J e D correspondem às intensidades das interações dipolares, ferromagnéticas e campo externo, respectivamente. O segundo termo refere-se a interação ferromagnética entre os átomos. Como foi descrito, os átomos dos ferromagnetos exercem e sofrem grande influência de seus vizinhos. Para simular essa interação em uma rede quadrada regular são considerados apenas os k átomos mais próximos (k=1,2,3,...,6).

O Hamiltoniano $\mathcal{H}$ descrito na (Eq.1) retorna a interação total entre todos os átomos do sistema em forma de energia. Essa energia é usada pra extrair algumas propriedades do modelo simulado, como: a) calor específico, b) variação média da energia e c) energia total do sistema. Análises físicas dos resultados numéricos obtidos a partir de modelos semelhantes ao descrito neste trabalho podem ser encontradas em (Morr et al., 1994) e (Ying et al., 2003).

Um dos objetivos do estudo de sistemas ferromagnéticos é observar seu comportamento em certas temperaturas. A partir de uma determinada temperatura, o sistema passa a apresentar diferentes características se comparado ao seu estado inicial. Essa temperatura é chamada de temperatura de Curie (CAO et al., 2007). É observado que, a partir dessa temperatura, a agitação atômica é tão intensa que os *spins* passam a ignorar a influência do campo externo e adotam direções aleatórias. Como descrito, esse comportamento molecular (ou atômico) pode ser observado em materiais paramagnéticos, portanto pode-se afirmar que essa temperatura corresponde à temperatura crítica do sistema, onde o sistema ferromagnético passa a se comportar como paramagnético.

3. Modelo Computacional

Ao longo do tempo, os *spins* dos átomos tendem a se alterar devido à diversos fatores. Com a finalidade de simular esse comportamento, nessa seção é descrito um modelo computacional baseado no método de Monte Carlo.

Um fator a ser considerado é a complexidade da resolução numérica da (Eq.1), em especial do primeiro termo (Eq.2) que reporta a interação dipolar nos materiais estudados. Por se tratar de um somatório duplo, portanto, um problema de todos para todos, pode-se atribuir à esse termo a complexidade computacional $\boldsymbol{O}(n^2)$, onde n é o número de *spins* no sistema.

3.1 Método de Monte Carlo

O método de Monte Carlo é um método estocástico usado para obter aproximações numéricas de funções complexas através da conversão de um modelo físico ou matemático em um modelo estatístico. O objetivo do método é prever o comportamento de um sistema através de amostragens retiradas de seu estado atual. A partir dessa idéia, Nicholas Metropolis e Stanislaw Ulam desenvolveram o algoritmo de Metropolis (Metropolis et al., 1953), muito usado para simular sistemas físicos. Detalhes sobre a implementação desse algoritmo podem ser encontrados em (Peçanha et al., 2008).

De maneira geral, o algoritmo de Monte Carlo Metropolis é descrito da seguinte forma: a cada iteração, um número k é sorteado aleatoriamente. A partir daí o k-ésimo átomo do sistema tem seu *spin* alterado. Para que esse *spin* seja aceito ele deve satisfazer a seguinte probabilidade:

$$W(\{S_k^a\} \rightarrow \{S_k^b\}) = \begin{cases} e^{(-\Delta\mathcal{H}/Kt)} & \Delta\mathcal{H} > 0 \\ 1 & \Delta\mathcal{H} \leq 0 \end{cases} \tag{3}$$

portanto, para que o sistema transite do estado a para o estado b e conseqüentemente o novo *spin* gerado para o átomo k seja aceito, a condição da (Eq.3) deve ser satisfeita, onde, $\Delta\mathcal{H} = \mathcal{H}_b - \mathcal{H}_a$ e $\mathcal{H}$ definido pela (Eq.1) (Yeomans, 1992), (Janke, 1998).

Uma etapa importante do algoritmo é a geração de números aleatórios. Para garantir que a probabilidade de sortear cada átomo seja uniforme, é utilizado o algoritmo *Mersenne Twister* cuja periodicidade é de 2^{19937} (Matsumoto & Nishimura, 1998). O átomo escolhido recebe um novo *spin* distribuído aleatoriamente no espaço de vetores unitários $\mathbb{S}^2$.

3.2 Paralelização do Modelo

O modelo físico proposto é constituído de um número praticamente infinito de átomos. Contudo, computacionalmente, o problema deve ser tratado de maneira discreta. Portanto, quanto maior o número de átomos, mais realistas serão os resultados obtidos. A principal desvantagem é que o tempo para se obter os resultados é proporcional ao tamanho do sistema. Neste trabalho estendemos nosso trabalho anterior (Peçanha et al., 2008) para permitir que a resolução computacional da (Eq.1) seja obtida em menor tempo e, conseqüentemente, que sistemas maiores possam ser simulados. Para atingir tal propósito, usamos um esquema de paralelização baseado no paradigma de memória compartilhada, em particular, utilizando múltiplos fluxos de execução (*multithreading*).

A energia de cada átomo pode ser computada de forma independente dos demais. Conseqüentemente, a energia referente a uma determinada partição do sistema também pode ser calculada de forma independente. Esta observação abre a possibilidade para que a computação da energia total do sistema seja dividida utilizando uma abordagem análoga ao modelo de computação mestre-escravo (Wilkinson & Allen, 1999). Um fluxo de execução principal poderia criar novos fluxos de execução (*threads*) que ficariam responsáveis por computar as energias de determinadas regiões do sistema.

Para dividir o trabalho entre os fluxos, utiliza-se um esquema de mapeamento dinâmico do espaço. A idéia é escolher planos alinhados com eixos, onde cada fluxo fica responsável por calcular as energias dos átomos localizados nesses planos. A energia do i-ésimo átomo é dada pela (Eq.1) e o somatório de todas as energias de interação de todos os átomos nos dá a energia total do sistema.

Resumidamente, nossa implementação segue os seguintes passos:

- Para N fluxos disparados, cada um fica responsável pelo cálculo da energia de um dos diferentes planos do espaço;

- Ao término da computação da energia de um determinado plano, o N-ésimo fluxo soma o seu resultado a uma variável global;
- Se ainda existirem planos a serem computados, um deles é direcionado a este N-ésimo fluxo, conforme ilustrado na figura 1;
- Ao término do cálculo da energia de todos os planos, a variável global com o somatório dos valores da energia de cada plano é retornada;
- Esses passos são seguidos a cada iteração.

Algumas restrições devem ser observadas para que a solução paralela funcione corretamente: a) fluxos diferentes devem computar energias de planos diferentes, b) a divisão dos planos deve seguir a uma das direções adotadas pelos eixos XY, YZ ou ZX e c) um plano cuja energia já tenha sido computada não deve ser acessado novamente.

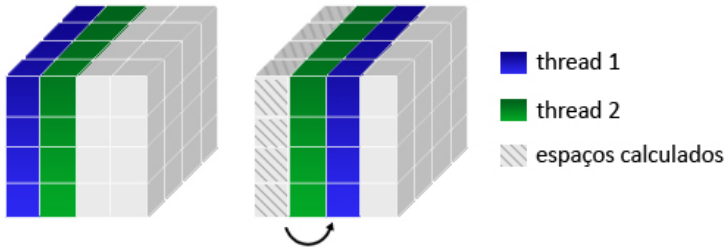

Figura 1: Na primeira figura, cada fluxo é responsável pelo cálculo da energia de um plano. Na segunda figura, o fluxo 1 (*thread 1*) termina o cálculo da energia do primeiro plano, busca um novo plano e marca o espaço já calculado como visitado.

O número de fluxos a ser utilizado na execução da aplicação pode variar. Em aplicações do tipo *CPU bound*, ou seja, que fazem uso intenso da CPU, costuma-se utilizar um número de fluxos igual ao número de processadores ou núcleos disponíveis na máquina. Deve-se ressaltar que um número de fluxos maior do que o número de processadores ou núcleos existentes na máquina pode ser escolhido, contudo, poderá haver perda de desempenho, já que haverá disputa entre os fluxos pelo uso dos processadores. Em um cenário ideal, o ganho de desempenho será proporcional ao número de processadores disponíveis, levando-nos a obter a chamada aceleração linear (*linear speedup*). Assim, no momento de se determinar a quantidade de fluxos que serão utilizados na execução, deve-se observar a quantidade de recursos disponíveis na máquina, e não o número de planos na qual o espaço de simulação foi dividido.

Poderia-se ainda adotar uma abordagem onde o espaço de simulação é previamente particionado. Contudo, descartamos esta abordagem, visto que o objeto a ser simulado pode adotar geometrias variadas e, caso uma destas seja assimétrica e as partições realizadas no espaço sejam regulares, células diferentes podem conter números de átomos diferentes. A discrepância entre o número de átomos por partição pode provocar desbalanceamento entre os fluxos, limitado o desempenho obtido. Isso decorre do fato de fluxos diferentes realizarem cálculos a mais (ou a menos) que outras, portanto, fluxos cujas tarefas já tenham sido realizadas permanecerão em espera até que todos os outros fluxos tenham finalizado suas tarefas.

A grande vantagem do modelo dinâmico implementado é que ele garante que um fluxo não ficará ocioso, a não ser que as energias de todos os planos já tenham sido calculadas. Uma das desvantagens observadas é o grande número de acessos a regiões de seção crítica.

Para implementar o modelo apresentado, usamos o conceito de *pool* de *threads*. Neste os fluxos são criados apenas uma vez a cada execução do programa. No início do algoritmo é indicado o número máximo de fluxos que o processo principal poderá disparar. Este número determinará o número de fluxos N no *pool*. Os fluxos ficam adormecidos até que alguma tarefa seja requisitada. Ao término dessa tarefa, o fluxo retorna ao *pool* e fica em estado de espera até que a solicitação de uma nova tarefa seja feita. O conceito de *pool* se contrapõe a implementação convencional, onde o fluxo é criado sob demanda e, ao término de seu trabalho, este é sincronizado e posteriormente destruído. O uso do *pool* de *threads* evita que fluxos sejam criados a todo momento, tornando o processo menos custoso.

4. Resultados

Nesta seção são apresentados os resultados obtidos nas simulações do modelo computacional descrito neste trabalho. As simulações foram executadas em uma máquina Dual Xeon E5410 Quad Core de 2.33Ghz com 4 GB de memória RAM, 12 MB de memória cache L2, placa-mãe Intel Workstation Board modelo S5000XVN e sistema operacional Windows Vista Ultimate 64 bits. Portanto 8 núcleos de processamento estão disponíveis nesta máquina.

A biblioteca *Pthread* (Butenhof, 1997) foi usada para implementar o modelo de paralelização descrito na seção anterior.

O modelo físico simulado consistiu de 729 átomos dispostos espacialmente em forma de cubo. O objetivo de nossas simulações é mostrar o fator de aceleração alcançado pelo modelo paralelizado, utilizando de 2 a 8 fluxos, com relação ao tempo de execução da sua versão seqüencial. O critério de parada adotado para a medição dos tempos foi o valor atribuído a energia média do sistema ao atingir seu estado de equilíbrio. O valor observado para o cubo de átomos usado é de -70,0.

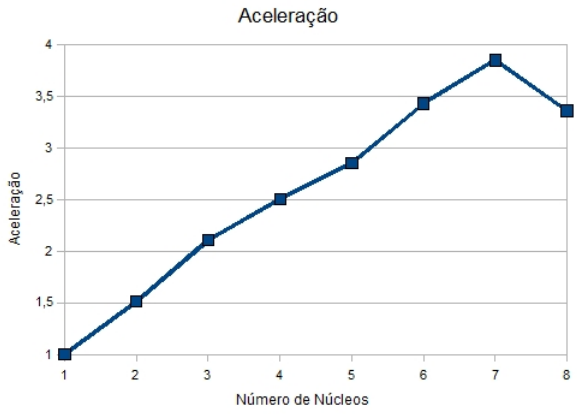

Figura 2: Aceleração média.

Pode-se perceber pela figura 2 que ocorre um aumento no desempenho do simulador quando sua versão paralela é utilizada. Como regra geral, quanto maior o número de fluxos envolvidos no processamento, maior é a aceleração obtida. No entanto, observa-se que esta aceleração não é linear, isto é, o fator encontrado não corresponde ao número de núcleos, e conseqüentemente, de fluxos em execução. Um dos fatores que influenciou nos resultados foi o escalonamento dos fluxos pelo sistema operacional. Apesar do mecanismo dinâmico para distribuição de carga entres os fluxos ter sido implementado, verificamos na prática a ocorrência de um desbalanceamento entre os fluxos: alguns fluxos ficaram sobrecarregados durante a simulação, recebendo mais tarefas do que outros fluxos.

Entretanto, o desbalanceamento de carga verificado, por si só, não explica completamente

o desempenho obtido. Acreditamos que uma contenção de memória possa estar ocorrendo. A contenção de memória ocorre quando dois ou mais núcleos ou processadores tentam acessar a memória simultaneamente. Como apenas um deles tem permissão para usar o barramento, os outros fluxos devem aguardar até que o mesmo seja liberado.

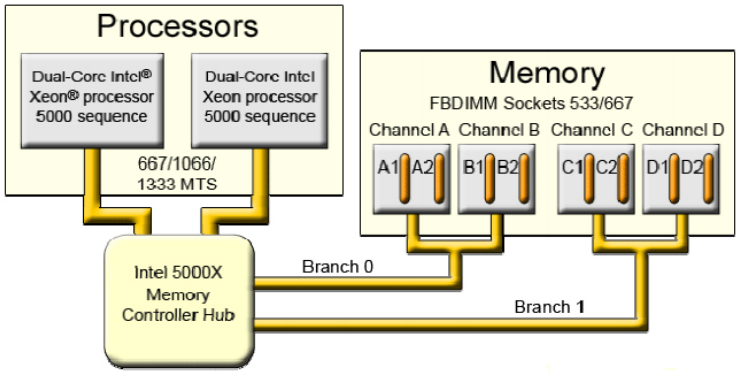

Figura 3: Arquitetura da placa Intel Workstation Board modelo S5000XVN (figura retirada de (Intel, 2009)).

O diagrama da figura 3 reproduz simplificadamente a disposição do *hardware* de memória e de processadores na placa-mãe da máquina de testes. As elipses denominadas A1, A2, B1, B2, C1, C2, D1 e D2 correspondem aos 8 *slots* de memória presentes. Note que existem dois barramentos, conectados e gerenciados pelo *hub* controlador de memória, que interligam as memórias aos processadores. Supondo que houvesse um fluxo para cada processador e dois módulos de memória, e que cada barramento acessasse somente um dos módulos, então não deveria haver contenção de memória. Porém, consta na especificação da fabricante (Intel, 2009) que os módulos de memória devam ser acoplados aos pares, preenchendo os *slots* na seqüencia A1 e B1, C1 e D1, A2 e B2, C2 e D2. O segundo barramento só é ativado quando os 8 slots estão completamente preenchidos. A máquina de testes possui dois módulos de 2GB nas posições A1 e B1. Logo os dois processadores precisam do mesmo barramento para que os fluxos operem sobre os dados necessários. Desta forma, um aumento no número de fluxos em execução acarreta em uma maior probabilidade de disputa pelo barramento. Deve-se ainda ressaltar que, adicionalmente, existem ainda duas competições internas a cada processador. A primeira delas é a competição de cada núcleo pelo barramento interno do processador que o liga ao barramento externo, causando assim uma contenção interna ao processador. Também existe, internamente, a competição de dois pares de núcleos pelo acesso a *cache* L2, que é compartilhada aos pares. Como os núcleos operam sobre conjuntos de dados diferentes, nem sempre estes dados estarão ocupando o mesmo bloco de memória; conseqüentemente, os núcleos não compartilharão os dados em *cache*. Logo, aumenta a probabilidade de ocorrência de *misses* na cache, o que por sua vez aumenta o número de vezes em que a memória principal é acessada para a obtenção dos bloco de dados.

Por fim, outro fator que pode ter influenciado o desempenho, em especial na configuração com 8 fluxos, é a disputa pela execução entre o SO e os seus processos e a aplicação. Com uma configuração com 7 fluxos, um núcleo fica livre, podendo ser utilizado para execução pelo SO. Com 8 fluxos criamos uma situação em que um núcleo é disputado pelo SO e a aplicação. Isso ajudaria a explicar a queda de desempenho observada quando passamos da configuração de 7 para 8 fluxos.

Ainda que os resultados obtidos não tenham se aproximado da desejável aceleração linear, deve-se destacar que foram responsáveis por uma substancial redução no tempo total de execu-

ção. Esta redução torna-se ainda mais importante a medida que um número maior de elementos é simulado, conforme podemos observar na figura 4.

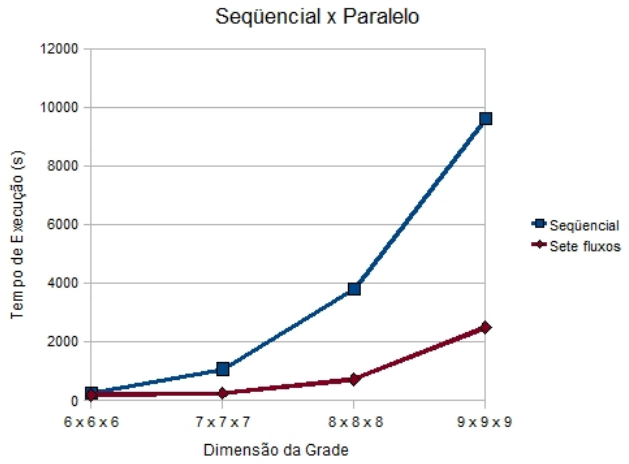

Figura 4: Comparação do tempo de execução das versões seqüencial e paralela, esta última com sete fluxos de execução.

5. Trabalhos Correlatos

Muitos trabalhos que simulam o comportamento de materiais ferromagnéticos podem ser encontrados na literatura. As diferenças entre este trabalho e os demais são a) o modelo físico empregado e b) a forma de tratar a complexidade do algoritmo que simula as interações nos materiais ferromagnéticos.

O modelo de Ising (Binder & Luijten, 2001) é um modelo mais simples para representar o comportamento de materiais ferromagnéticos. Neste modelo, o *spin* de um átomo i admite apenas dois valores: $S_i = \pm 1$, o que impõe ao vetor de *spins* uma restrição de alinhamento paralelo à direção da quantização introduzida pelo campo magnético. Existem sistemas (Yeomans, 1992) onde o modelo de Ising não pode ser empregado e outros onde a restrição que o caracteriza reduz o grau de realismo dos resultados. O modelo de *spins* descrito nesse trabalho, o de Heisenberg, oferece um maior realismo e trata o problema da simulação de ferromagnetos de uma forma menos restritiva.

Abordagens onde *clusters* de *spins* são alterados simultaneamente em um único passo de Monte Carlo são estudados há algum tempo. Swendsen e Wang (Swendsen & Wang, 1987) foram os precursores dessa idéia. A partir dessa proposta, Luijten e Blöte (Luijten & Blöte, 1995) conseguiram reduzir o número de operações por passo de Monte Carlo drasticamente para $\boldsymbol{O}(N \log N)$. Recentemente Fukui e Todo obtiveram um expressivo resultado ao propor um método onde o tempo de execução chegou a ser proporcional à $\boldsymbol{O}(N)$ (Fukui & Todo, 2009). Vale ressaltar que os Hamiltonianos dos trabalhos citados podem ser paralelizados utilizando o algoritmo proposto nesse trabalho.

Em (Santos et al., 2008) podem ser encontrados alguns algoritmos que paralelizam o método de Monte Carlo aplicado a sistemas ferromagnéticos. Contudo, a abordagem é diferente da proposta neste trabalho. O foco do paralelismo está no passo de Monte Carlo. Após um número grande de iterações, o sistema tende a se estabilizar, com isso existe uma dificuldade natural de se gerar vetores que satisfaçam a (Eq.3). Isso justifica o uso do paralelismo, com o objetivo de evitar que os cálculos realizados ($\Delta\mathcal{H}$, necessário para validar o *spin*) não sejam aproveitados na simulação. Isso é observado pois, caso a condição (Eq.3) não seja aceita para um vetor de

spin gerado, os cálculos realizados no último passo são descartados e o sistema retorna ao seu estado anterior.

6. Conclusão

Neste trabalho apresentamos o processo de paralelização do MCSE (*Monte Carlo Spin Engine*), um simulador de *spins* em estruturas tridimensionais genéricas formadas por átomos com propriedades magnéticas. O processo de paralelização se fez necessário pela observação de que a simulação dos modelos envolve um alto custo computacional, diretamente associado à quantidade de elementos presentes no sistema simulado, o que, em última instância, limita o tamanho do sistema simulado.

A primeira tentativa de redução dos custos computacionais utilizando vários fluxos de execução reduziu significativamente o tempo total de execução. Em particular, com 7 fluxos de execução verificamos uma aceleração igual a 3,85.

Apesar dos resultados positivos, acreditamos que existe ainda margem para um maior ganho de desempenho. Especificamente, gostaríamos de investigar melhor o possível problema da contenção de memória, buscando alternativas para minorá-lo, como um melhor mapeamento dos dados na memória, a fixação da execução dos fluxos em um determinado núcleo. Outra problema que pretendemos atacar futuramente é o desbalanceamento de carga observado na prática. Estamos estudando a viabilidade de adicionar estruturas que adéqüem o balanceamento de carga entre os fluxos sem aumentar a complexidade do algoritmo como, por exemplo, utilizando *Octrees*.

Por fim, estamos iniciando a implementação de uma versão paralela do simulador empregando GPGPUs (*General-Purpose computation on Graphics Processing Units*). GPGPUs são placas de processamento gráfico altamente paralelas, e que podem ser programadas, oferecendo hoje capacidade de processamento muitas vezes superior a capacidade de processamento obtida com CPUs.

Agradecimentos

Os autores agradecem à Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES), ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), a UFJF e à Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG) pelo suporte dado ao desenvolvimento deste trabalho.

REFERÊNCIAS

Binder, K. & Luijten, E., 2001. Monte carlo tests of renormalization-group predictions for critical phenomena in ising models. *Physics Reports*, vol. 344, n. 4-6, pp. 179 – 253.

Butenhof, D. R., 1997. *Programming with POSIX Threads*. Addison-Wesley Reading, MA, USA, 1 edition.

CAO, L., XIE, D., xing GUO, M., Park, H., & Fujita, T., 2007. Size and shape effects on curie temperature of ferromagnetic nanoparticles. *Transactions of Nonferrous Metals Society of China*, vol. 17, n. 6, pp. 1451 – 1455.

dos Santos Cabral Neto, J., 2004. *Estudo Sistemático das Propriedades Termodinâmicas e criticalidade de Filmes Finos e Super-Redes Magnéticas*. Universidade Federal de São Carlos.

Fukui, K. & Todo, S., 2009. Order-n cluster monte carlo method for spin systems with long-range interactions. *Journal of Computational Physics*, vol. 228, n. 7, pp. 2629 – 2642.

Intel, 2009. *Intel Workstation Board S5000XVN: Technical Product Specification*. Enterprise Plataforms and Services Division - Marketing.

Janke, W., 1998. Nonlocal monte carlo algorithms for statistical physics applications. *Mathematics and Computers in Simulation*, vol. 47, n. 2-5, pp. 329 – 346.

Landau, D. & Binder, K., 2005. *A Guide to Monte Carlo Simulations in Statistical Physics*. Cambridge University Press, New York, NY, USA.

Luijten, E. & Blöte, H. W. J., 1995. Monte carlo method for spin models with long-range interactions. *Int. J. Mod. Phys. C*, vol. 6, pp. 359–370.

Matsumoto, M. & Nishimura, T., 1998. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul.*, vol. 8, n. 1, pp. 3–30.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., & Teller, E., 1953. Equation of state calculations by fast computing machines. *Journal of Chemical Physics*, vol. 21, pp. 1087–1092.

Morr, D., Jensen, P., & Bennemann, K.-H., 1994. Reorientation transition of the magnetization in thin ferromagnetic films. *Surface Science*, vol. 307-309, n. Part 2, pp. 1109 – 1113. Proceedings of the European Conference on Surface Science.

Peçanha, J., Campos, A., Pampanelli, P., Lobosco, M., Vieira, M., & Dantas, S., 2008. Um modelo computacional para simulação de interação de spins em elementos e compostos magnéticos. *XI Encontro de Modelagem Computacional*, vol. .

Santos, E. E., Rickman, J. M., Muthukrishnan, G., & Feng, S., 2008. Efficient algorithms for parallelizing monte carlo simulations for 2d ising spin models. *J. Supercomput.*, vol. 44, n. 3, pp. 274–290.

Swendsen, R. H. & Wang, J.-S., 1987. Nonuniversal critical dynamics in monte carlo simulations. *Physical Review Letters*, vol. 58, n. 2, pp. 86+.

Wilkinson, B. & Allen, M., 1999. *Parallel programming: techniques and applications using networked workstations and parallel computers*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Yeomans, J. M., 1992. *Statistical Mechanics of Phase Transitions*. Oxford University Press.

Ying, L., Nanxian, C., Hongmin, Z., & Chengwen, W., 2003. Monte carlo simulation of the reorientation transition in heisenberg model with dipole interactions. *Solid State Communications*, vol. 126, n. 4, pp. 223 – 227.

COMPUTATIONAL SIMULATION OF SPINS INTERACTION IN MAGNETIC SYSTEMS USING MULTITHREADING

Abstract. *The study of spins interaction in magnetic compounds and in magnetic elements is an important research area. It also leads to a better understanding of magnetic properties of matter. An approach to the study of spins interaction is the use of physical models, which can be simulated computationally. However, we should consider the complexity involved in numerical solution of physical models. This is usually related to the number of elements in the simulated structure. Therefore, the development of computational solutions that contribute to reduce the processing time is of utmost importance. This paper presents a physical model that describes the behavior of a special class of materials, the ferromagnetic. A multithread-based parallel computational model, which aims to reduce the costs related to the numerical solution, is also presented. The preliminary results show a reduction of up to 3.85 times in execution time when compared to the sequential version.*

Keywords: *Physical simulation, Spins interactions, Simulation of ferromagnetic materials, Parallelization of physical models*