14° POSMEC - Simpósio do Programa de Pós-Graduação em Engenharia Mecânica

Universidade Federal de Uberlândia
Faculdade de Engenharia Mecânica

SIMULAÇÃO NUMÉRICA DA CONDUÇÃO DE CALOR TRIDIMENSIONAL EM UM CLUSTER BEOWULF

Wellington Pereira Marinho
Faculdade de Engenharia Mecânica, Universidade Federal de Uberlândia – Uberlândia, MG 38400-902 – Brasil.
Laboratório de Transferência de Calor e Massa e Dinâmica dos Fluidos.
wellmarinho@yahoo.com.br

Rubens Campregher*
campregher@mecanica.ufu.br

Aristeu da Silveira Neto*
aristeus@mecanica.ufu.br

Resumo: *O presente trabalho apresenta os recursos envolvidos na implementação de um cluster classe Beowulf para resolver problemas de transferência de calor tridimensional. O fenômeno simulado refere-se à condução de calor no interior de um sólido homogêneo e isotrópico, onde o domínio computacional - discretizado por diferenças finitas - pode ser dividido entre vários processadores em um ambiente distribuído. Além disso, são analisados tópicos importantes em processamento paralelo como características de hardware e software, decomposição do domínio, balanceamento de carga, problemas nas regiões de interface entre os subdomínios, troca de informações nas fronteiras, speedup e eficiência. Os resultados de speedup e eficiência são explorados para avaliar a performance do programa paralelo quando comparado a um programa serial. A curva de speedup é obtida a partir de um problema de tamanho fixo onde o número de processadores foi sendo variado, medindo-se o tempo para cada configuração. Além disso, a sua análise auxilia na determinação do número máximo de processadores possível de ser utilizado em um determinado programa paralelizado, otimizando o uso dos recursos computacionais. O trabalho mostra, também, o comportamento da curva de speedup em função do refinamento da malha e a importante relação da granularidade com a eficiência do software. O código computacional foi escrito em linguagem C com biblioteca de paralelização MPI.*

Palavras-chave: *Cluster Beowulf, programação paralela, transferência de calor, speedup, eficiência*

1. INTRODUÇÃO

Um dos grandes problemas envolvendo simulação de problemas físicos complexos é a disponibilidade de recursos computacionais, uma vez que investimentos em computadores de última geração geralmente são inviáveis. Dessa forma, a implementação de um cluster Beowulf caseiro torna-se a mais apropriada, dado o seu baixo custo quando comparado aos supercomputadores disponíveis comercialmente.

Um cluster é um sistema que compreende dois ou mais computadores ou sistemas (denominados nós) os quais trabalham em conjunto para executar aplicações ou realizar outras tarefas, de tal forma que os usuários tenham a impressão da existência de um único sistema, criando assim uma ilusão de um recurso único (Pitanga, 2002).

O fenômeno simulado neste trabalho refere-se à condução de calor em um sólido, onde o domínio computacional é dividido entre vários processadores. O conhecimento obtido aqui ajudaria na implementação de um "solver" paralelo para a equação de convecção-difusão de quantidade de movimento ou outras propriedades.

2. CLUSTERS BEOWULF

Os modelos de arquitetura baseados em clusters têm como uma das suas formas mais populares os chamados clusters Beowulf. Como definição, pode-se dizer que esta configuração é uma arquitetura de multicomputadores utilizada para computação paralela, compondo um conjunto de máquinas distribuídas na forma de um nó servidor e nós clientes conectados via rede (Ethernet ou outra topologia qualquer), rodando um sistema operacional paralelo que permita a implementação de estruturas de Máquinas Virtuais Paralelas (PVM - Parallel Virtual Machines) e/ou Interface de Passagem de Mensagens (MPI - Message Passing Interface). O servidor tem a função de controlar todo o cluster, distribuir os arquivos e as tarefas para os nós clientes e servir de porta de entrada e saída para conexão com uma rede externa, se for o caso. Pode também haver mais de um servidor em um cluster Beowulf, podendo ser dedicado à operações específicas como monitoração ou comunicação com o exterior, ou, ainda, para servir como console apenas. Cada nó de um cluster Beowulf pode ser o mais simples possível, sem interfaces de vídeo ou áudio, unidades de disco removível, portas paralelas, etc. Todos os nós clientes são configurados a partir do servidor e só desenvolvem tarefas que lhes são determinadas por ele.

O nome Beowulf foi tomado emprestado de um dos mais antigos poemas épicos da língua inglesa, que conta a história de um herói, de grande força e valentia, em sua saga para derrotar o monstro de Grendel. O primeiro cluster Beowulf foi desenvolvido em 1994 por Thomas Sterling e Don Becker (Pitanga, 2002), pesquisadores do Centro de Excelência em Dados Espaciais e Informações Científicas (CESDIS) da Agência Espacial Americana (NASA) no Centro Espacial Goddard em Greenbelt, Maryland. Foi implementado para utilização no projeto de Ciências Espaciais e Terrestres (ESS) em conjunto com o grupo de Computação e Comunicação de Alta Performance (HPCC) do qual os pesquisadores faziam parte.

2.1 Características de um Cluster Beowulf

Um cluster Beowulf não é um pacote de software ou uma nova topologia de rede, um hardware especial e nem um sistema operacional com kernel diferenciado (Gottlieb, 2001): é uma maneira de interligar estações que rodam Linux (ou algum outro sistema operacional paralelo) de forma a simular o comportamento e a performance de um supercomputador paralelo. Embora existam muitos pacotes de software, versões de kernel, bibliotecas PVM e MPI diversas e aplicativos de configuração que fazem a arquitetura de cluster mais rápida ou fácil de configurar, é possível construir um Beowulf com qualquer uma das distribuições atuais padrão do Linux e estações de trabalho comuns. Basicamente, uma configuração de duas ou mais máquinas rodando Linux em rede, compartilhando pelo menos uma estrutura de diretórios via NFS, capacidade de executar shell remoto (rsh) e bibliotecas de PVM ou MPI, já se constitui um cluster Beowulf.

Na taxonomia tradicional de computação paralela, um cluster Beowulf pode ser situado como o meio-caminho entre as máquinas de processamento paralelo massivo (MPPs - Massive Parallel Processors), como o Cray ou os Hipercubos, e uma rede de estações (NOW - network of workstations), com características e benefícios de ambas. As MPPs são máquinas tipicamente maiores, mais caras e mais rápidas em sua conexão interna que os clusters. A programação é complexa, pois é preciso preocupar-se com detalhes como localidade dos processos, balanceamento de carga, granularidade e overhead na comunicação para atingir a performance ótima, dentre outras. Para a programação em uma rede de estações são necessários algoritmos bastante tolerantes quanto à problemas de distribuição da carga e latência provocada pela forma de comunicação. Para um cluster

Beowulf, qualquer programa - tanto para MPPs como para NOWs - que não necessite de uma granularidade muito fina de processamento pode ser portado sem grandes dificuldades.

Um cluster Beowulf se diferencia de uma rede de estações (NOW) por algumas poucas, porém importantes, características. Em primeiro lugar, os nós de um cluster são dedicados exclusivamente ao processamento interno do cluster. Isso facilita no balanceamento de carga, já que a performance de cada nó não varia por força de fatores externos, uma vez que estão isolados do mundo exterior, rodando somente as aplicações determinadas pelo servidor. Em segundo lugar, nas redes de estações, além de existirem outras informações e processos que não pertencem ao domínio do programa que está sendo executado em paralelo e que podem diminuir a eficiência, também podem gerar um problema de segurança. Em um cluster todos os processos que rodam em qualquer nó são controlados através de um número de identificação gerado e controlado pelo servidor, não havendo portanto chance de execução de processos de atividade desconhecida. Finalmente, em uma rede de estações é preocupação do sistema operacional tornar a máquina mais amigável ao usuário, ocasionando desperdício de recursos que, no caso do cluster serão aproveitados para o processamento, já que os nós clientes não têm interação direta com o usuário.

2.2 Componentes de um Cluster

2.2.1 Hardware

Um cluster Beowulf é construído a partir de máquinas de modelos comuns, amplamente disponíveis comercialmente, sendo este um de seus pontos fortes. A sua construção não requer nenhum hardware especial nem apresenta dificuldades técnicas adicionais de instalação/implementação. A título de exemplo, um conjunto de máquinas formado por oito estações comuns de trabalho (preferencialmente semelhantes) com placa de rede já é o suficiente para se iniciar a implementação de um cluster Beowulf.

Este trabalho foi realizado em um cluster homogêneo de cinco máquinas, isto é, todos os seus nós possuem as mesmas capacidades de memória e processamento e a mesma rede de comunicação interligando-os. O nó mestre é composto por uma placa-mãe Intel® 865PERL, processador Pentium4® 2.8GHz, memória RAM DDR 1024 MB, HD IDE 80 GB e placa de vídeo Radeon® 9200 128 MB DDR AGP8x. Como um cluster homogêneo, os nós escravos têm a mesma configuração, exceto pela placa de vídeo, sendo uma Geforce® 64 MB AGP4x – a tarefa de pós-processamento são realizadas pelo nó mestre. Todas as máquinas são conectadas por uma rede Gigabit Ethernet utilizando um switch 3COM de 8 portas. A arquitetura do cluster pode ser vista na Figura 1.

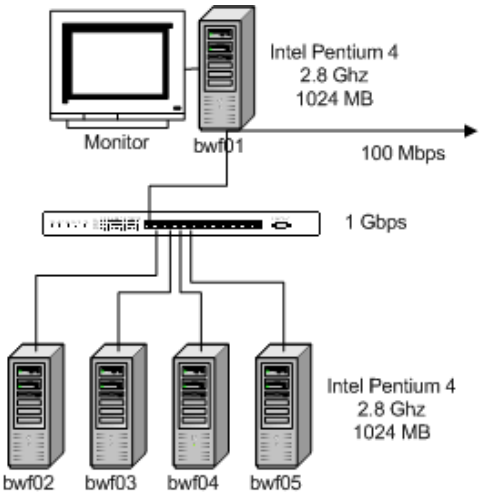

Figura 1: Arquitetura do Cluster

2.2.1 Software

O cluster roda em sistema operacional Linux Red Hat® 9.0, escolhido por apresentar um fácil gerenciamento e disponibilidade de atualização (Sonzogni *et al*., 2002). Para o gerenciamento da rede tem sido utilizada a ferramenta bWatch® que fornece informações sobre conectividade, memória e swap e para pós-processamento são utilizados Paraview® e Opendx®.

Para a distribuição de tarefas entre os nós do cluster foi escolhida a biblioteca MPI (*Message Passing Interface*). A MPI é um padrão de comunicação desenvolvido por cerca de 60 pessoas em 40 organizações, a maioria deles dos Estados Unidos. Muitos fabricantes de computadores estão envolvidos no projeto, como também, institutos de pesquisa, indústria e laboratórios governamentais. A "filosofia" por trás do projeto MPI é promover encontros freqüentes com a finalidade de incluir novas características a serem testadas e implementadas por usuários ao redor do mundo.

3. MODELO MATEMÁTICO.

Um problema de condução de calor tridimensional foi considerado para ilustrar a estratégia de paralelização e análise de performance. A equação de transporte de energia (Equação 1), de onde se obtém o campo de temperatura T(x,y,z,t), é dada por:

$$\frac{\partial(\rho T)}{\partial t} = \frac{\partial}{\partial x}\left(\frac{k}{c_p}\frac{\partial T}{\partial x}\right) + \frac{\partial}{\partial y}\left(\frac{k}{c_p}\frac{\partial T}{\partial y}\right) + \frac{\partial}{\partial z}\left(\frac{k}{c_p}\frac{\partial T}{\partial z}\right) + S \qquad (1)$$

onde ρ é a densidade do sólido, k é a condutividade térmica, c_p é o calor específico e S é o termo fonte.

O domínio foi discretizado pelo método de diferenças finitas (Fortuna, 2000). O domínio numérico pode ser visto na Figura 2.

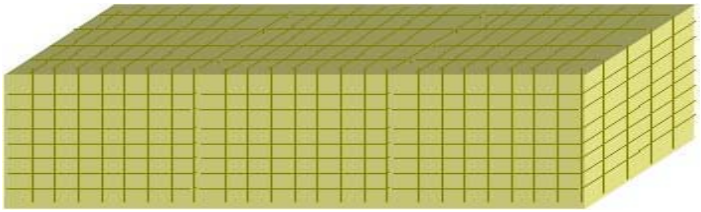

Figura 2: Domínio numérico

O sistema de equações algébricas Equação 2, derivado da discretização numérica foi resolvido usando um SOR (*sucessive over-relaxation*) paralelo como descrito abaixo (o índice *k* sobrescrito representa a iteração corrente):

$$\phi_{i,j,k}^{k+1} = \frac{\omega\left(A_L\phi_{i+1,j,k}^{k} + A_W\phi_{i-1,j,k}^{k+1} + A_N\phi_{i,j+1,k}^{k} + A_S\phi_{i,j-1,k}^{k+1} + A_T\phi_{i,j,k+1}^{k} + A_B\phi_{i,j,k-1}^{k+1} + S_{i,j,k}\right)}{A_P} + (1-\omega)\phi_{i,j,k}^{k} \qquad (2)$$

As condições de contorno para o domínio 3D são do tipo Dirichlet nas paredes laterais e Neumann nas paredes superior e inferior, como pode ser visto na Figura 3. As condições iniciais no interior do domínio e nas faces em y foram 0°C. As faces restantes foram iniciadas com 150°C. Ao longo da simulação as temperaturas nas faces x e z são mantidas a 150ºC e as faces em y foram

consideradas isoladas. Os resultados para um domínio com dimensões Lx, Ly, Lz com 240 X 30 X 90 pontos, respectivamente, são descritas na Figura 7.

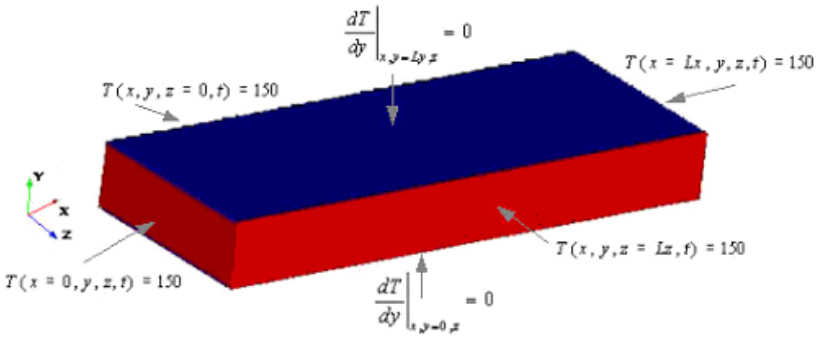

Figura 3: Condições de contorno do domínio numérico

1. Um aspecto importante na implementação de um programa paralelo é a decomposição do domínio, especialmente no que diz respeito ao balanceamento de carga, onde subdomínios de tamanhos iguais são desejáveis. A decomposição resulta em um certo número de tarefas, cada qual com seu conjunto de dados podendo ser feita de diferentes maneiras, dependendo da estrutura de dados (Pacheco, 1996). A Figura 4 mostra três formas diferentes de decomposição para um domínio computacional 3D.

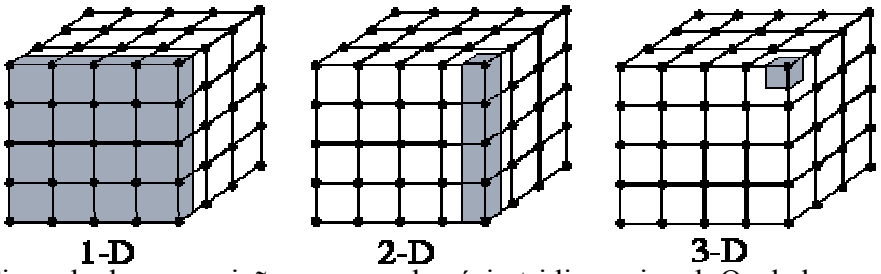

Figura 4: Tipos de decomposição para um domínio tridimensional. Os dados associados a uma tarefa estão sombreados.

A decomposição em uma, duas ou três dimensões é possível para uma malha tridimensional. Para este trabalho inicial, o problema de condução foi decomposto em uma dimensão por apresentar uma fácil implementação no processo de comunicação e sincronização. A decomposição do domínio está ilustrada na Figura 5.

Apesar de o domínio estar dividido entre os processos disponíveis, estes não são independentes entre si para a maioria dos problemas numéricos. Neste caso, cálculos executados em cada subdomínio requerem dados pertencentes a subdomínios adjacentes. Dessa forma, dados devem ser trocados entre os processadores a fim de atualizar as condições de contorno.

A Figura 6 ilustra o processo de troca de mensagens onde somente um plano é sobreposto, sendo o mínimo requerido em uma aproximação de segunda ordem. É importante notar que as condições de contorno físicas ao longo do eixo *x* (tipo Dirichlet) são aplicadas somente ao primeiro e último processador, isto é, na face x = 0 no subdomínio P0 e na face x = Lx no subdomínio PN. As condições de contorno requeridas para resolver o problema nos subdomínios restantes são obtidas através da troca de dados descrita acima. Ao longo dos eixos *y* e *z*, as condições de contorno físicas são exatamente as mesmas para todos os subdomínios sendo que a troca de dados não é necessária, visto a utilização do método de decomposição em uma dimensão.

O processo de sincronização é uma das tarefas mais difíceis na implementação de algoritmos paralelos. A sincronização requer uma combinação estrita entre envio e recebimento de mensagens.

De fato, algoritmos bem paralelizados devem fornecer resultados iguais ao algoritmo serial dentro da precisão estabelecida.

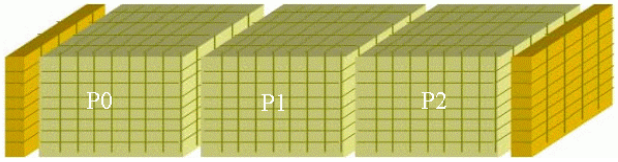

Figura 5: Decomposição em uma dimensão de um domínio tridimensional.

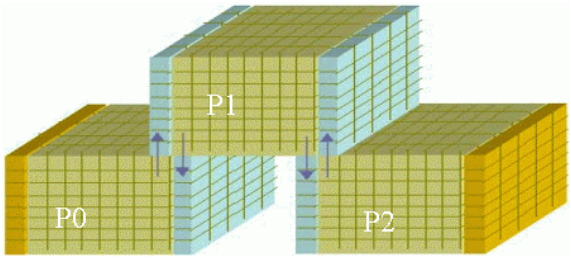

Figura 6: "Overlapping" e troca de mensagens entre os subdomínios.

4. RESULTADOS E DISCUSSÃO

Primeiramente, os resultados paralelos foram comparados com o serial sobre as mesmas condições de contorno iniciais. A Figura 7 representa o campo temperatura no plano y = Ly/2, obtido para simulações de um (N = 1) a cinco (N = 5) processadores. Além disso, para cada campo de temperatura abaixo, foi extraído um perfil de temperatura de 40 pontos ao longo do eixo *x*, tornando possível quantificar os resultados.

Nota-se que em todas as simulações, o campo de temperatura é o mesmo dentro da tolerância do *solver*, isto é, 1.0×10^{-6}. Esta característica denota consistência do algoritmo paralelo quando comparado ao programa serial. A Fig (8) mostra a evolução no tempo do campo de temperatura para a malha de 240 x 30 x 90, empregando 5 processadores. Há cinco tomadas de tempo diferentes, de t_0 = 2s a t_4 = 10s, com passo de tempo de 2s.

Para avaliar a performance do algoritmo paralelo são considerados, principalmente, dois conceitos básicos (Baker & Smith, 1996), quais sejam, o *speedup* e a eficiência. O primeiro é definido como:

$$S(N) = T(1)/T(N) \tag{3}$$

onde T(1) é o tempo obtido com apenas um processador, T(N) é o tempo requerido para resolver o mesmo problema em uma arquitetura paralela tendo um número N de processadores. A eficiência, que pode ser entendida como a fração de tempo gasto dos processadores fazendo o trabalho útil, é definida por:

$$E(N) = S(N)/N \tag{4}$$

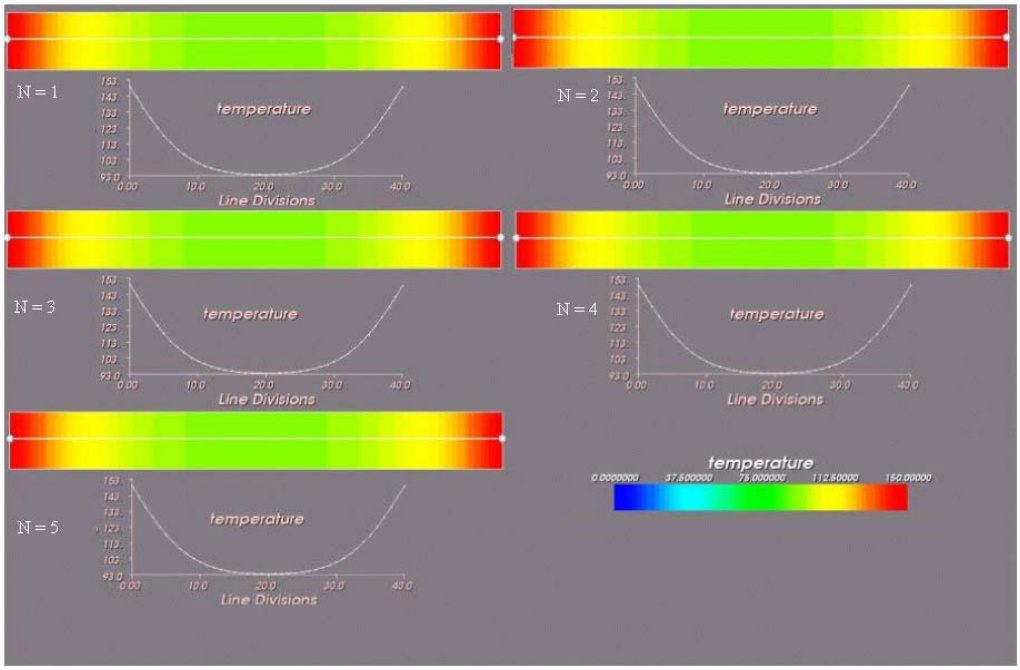

Figura 7: Campo de temperatura no plano xz com o número de processadores variando de um (N = 1) a cinco (N = 5).

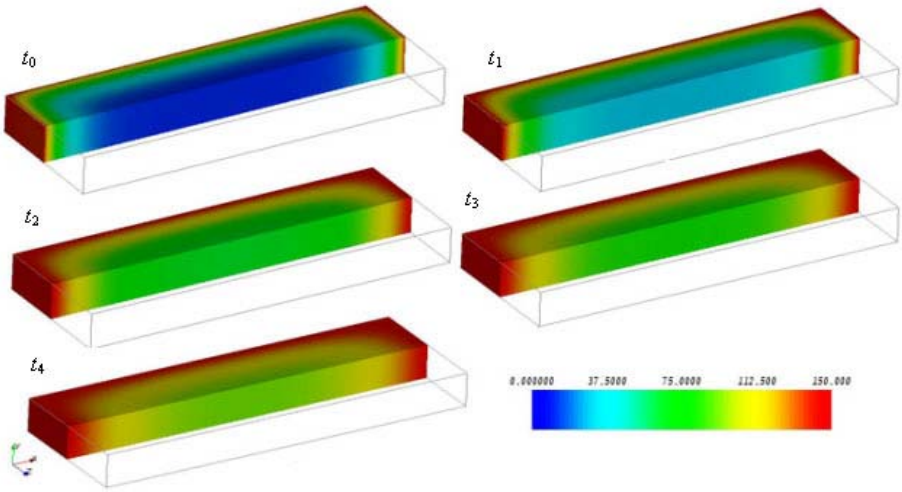

Figura 8: Evolução do campo de temperatura no tempo, de t = 2 a t = 10 segundos.

A Tabela 1 representa o "speedup" e a eficiência para os resultados obtidos rodando em apenas um processador ao número máximo de processadores, isto é, usando cinco máquinas. Na Figura 9, os resultados para o *speedup* são ilustrados graficamente e comparados com o valor ideal, representado pela curva azul. A Figura 10 mostra a eficiência obtida para a mesma malha da Figura 9. A eficiência ótima corresponde a E = 1.0 (ou 100%).

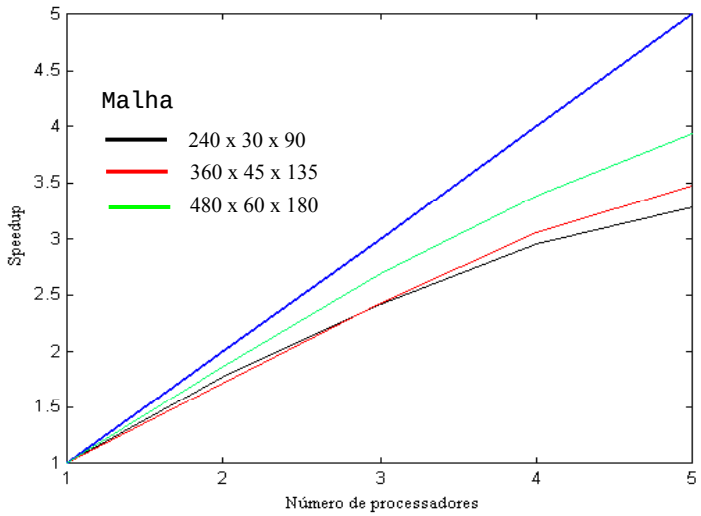

Figura 9: Resultados de *speedup* para três malhas de tamanhos diferentes. A curva azul representa o ideal.

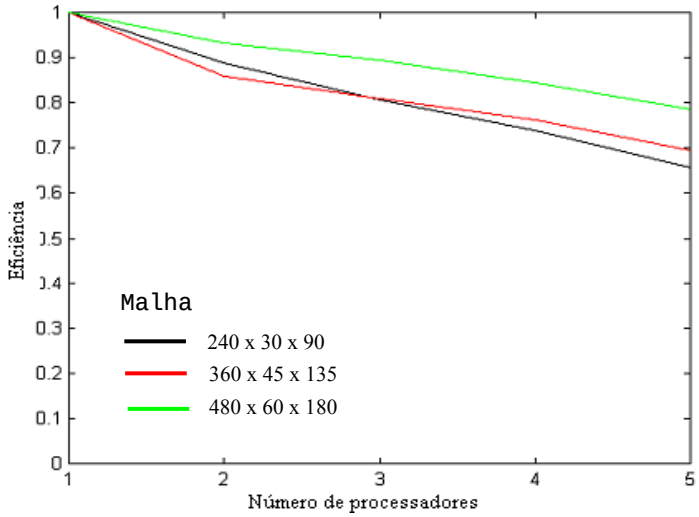

Figura 10: Resultados de eficiência para três malhas de tamanhos diferentes.

Pode-se notar que com cinco processadores, o *speedup* aumenta com o tamanho da malha. Esta característica é importante para avaliar o *speedup* ótimo do algoritmo, ou seja, ajuda a determinar o número de processadores em que o *speedup* é máximo. Os resultados mostram que este número ainda não foi atingido, sendo possível adicionar mais processadores sem redução do desempenho.. Outra observação refere-se ao tamanho da malha. Quanto mais refinada a malha se torna, melhor é a curva do *speedup*. A justificativa para este comportamento é o fato de que o tempo consumido no processamento de cálculos, principalmente solução do sistema linear, torna-se relativamente maior quando comparado ao tempo gasto na transferência de dados entre os processadores. Para malhas grosseiras, o tempo gasto na troca de informações tende a superar o tempo da solução do problema. A Figura 10 pode ilustrar a idéia do "trabalho útil" derivada da definição de eficiência onde a relação entre o tempo de processamento e o tempo de comunicação é mais favorável para malhas mais refinadas. Grosseiramente falando, para uma malha de 460 x 60 x 180 rodando com cinco processadores, aproximadamente 80% do tempo de simulação é gasto no processamento e o restante do tempo gasto na comunicação. Uma análise similar pode ser feita para outros tamanhos de malha. Apesar destes resultados serem interessantes, eles são específicos para este *hardware* (máquinas e estrutura da rede) e *software* (algoritmo, biblioteca de passagem de mensagens, sistema operacional, etc).

Tabela 1: Resultados de *speedups* e eficiência para um número de um até cinco processadores.

Tamanho da malha	N. de processadores	Speedup	Eficiência (%)
240x30x90	1	1,00	100,0
	2	1,78	88,9
	3	2,42	80,5
	4	2,95	73,8
	5	3.28	65,6
360 x 45 x 135	1	1,00	100,0
	2	1,72	85,9
	3	2,43	80,9
	4	3,05	76,2
	5	3,48	69,6
480 x 60 x 180	1	1,00	100,0
	2	1,87	93,4
	3	2,69	89,5
	4	3,38	84,4
	5	3,93	78,6

5. CONCLUSÃO

Beowulf é um projeto bem sucedido. A opção feita por seus idealizadores de usar hardware popular e software aberto tornou-o fácil de usar e manter. O aumento do número de clusters *Beowulf* não nega isto. Mais do que um experimento, foi obtido um sistema de uso prático que continua sendo melhorado. A recente incorporação aos microprocessadores de características antes somente encontradas nos supercomputadores, a baixo custo, torna viável usá-los em sistemas multiprocessados que não dependam tanto da rapidez do meio de interconexão. Para que seja possível obter proveito desse tipo de sistema, é necessário aumentar a granularidade da aplicação, de modo a fazer o mínimo possível de movimentação de dados entre os subsistemas e o máximo possível de processamento dentro de cada um deles.

6. REFERÊNCIAS

Baker, L. and Smith, B.J., 1996, "Parallel Programing", Computing McGraw-Hill.

Fortuna, A. O., 2000, "Técnicas Computacionais Para Dinâmica Dos Fluidos: Conceitos Básicos e Aplicações"

Gotlieb, S., 2001, "Cost-Effective Clustering", Computer Physics Communications, Vol.142, pp. 43-48.

Pitanga, M., 2002, "Construindo supercomputadores em Linux", Ed. Brasport.

Pacheco, P.,1996, "Parallel Programming with MPI", Morgan Kaufmann.

Sonzoni,V.E., Yommi, A.M., Nigro,N.M., Storti, M.A., 2002, "A parallel finite element program on a Beowulf cluster", Advances in Engineering Software, Vol. 33, pp. 427-433

7. DIREITOS AUTORAIS

Os autores são os únicos responsáveis pelo conteúdo do material impresso incluído no seu trabalho.

THREE-DIMENSIONAL HEAT TRANSFER SIMULATION IN A BEOWULF CLUSTER

Wellington Pereira Marinho
Faculdade de Engenharia Mecânica, Universidade Federal de Uberlândia – Uberlândia, MG 38400-902 – Brasil.
Laboratório de Transferência de Calor e Massa e Dinâmica dos Fluidos.
wellmarinho@yahoo.com.br

Rubens Campregher*
campregher@mecanica.ufu.br

Aristeu da Silveira Neto*
aristeus@mecanica.ufu.br

***Abstract*:** *This work presents the resources involved in the assembling of a Beowulf-class cluster to solve three-dimensional heat transfer problems. The numerical problem refers to heat conduction inside a homogenous isotropic solid, which its computational domain – discretized by the finite-difference method – can be split among several processors into memory distributed architecture. Furthermore, several important topics in parallel processing are discussed, such as hardware and software characteristics, domain decomposition, load balancing, sub-domains interface problems, interface data exchange, speedup, and efficiency. The speedup and efficiency results are explored in order to evaluate the parallel program performance when compared to a serial one. The speedup curve is obtained from a fixed size domain which the processor number was gradually increased. The analysis of this parameter helps find the ideal number of processors for each parallel code. It is possible, also, observe the speedup curve behavior in function of the mesh refinement as well as the relation between granularity and software efficiency. The computational code was written in C with MPI parallel library.*

Keywords: Beowulf cluster, parallel program, heat transfer, speedup, parallel efficiency.