Utilização da Computação Paralela Distribuída na Otimização do Processamento de Imagens Médicas Fazendo Uso do JPVM

Ricardo J. Sabatine, Priscila T. M. Saito, Fátima L. S. Nunes, Kalinka R. L. J. C. Branco

UNIVEM – Centro Universitário "Eurípides Soares da Rocha" de Marília
Marília – São Paulo – Brasil – Caixa Postal 2041
{sabatine, psaito, fatima, kalinka}@univem.edu.br

Resumo

Este artigo tem como objetivo demonstrar a viabilidade da otimização do tempo do processamento de imagens, mais especificamente de imagens médicas, sem que ocorra a perda de *pixels* da imagem através do uso biblioteca de passagem de mensagem JPVM a qual tem como hospedeira a linguagem de programação Java.

1. Introdução

O processamento de imagens tem se mostrado um campo importante de pesquisa científica e tecnológica sendo utilizado nas mais diversas áreas como cinematográfica, meteorológica entre outras [5]. Na medicina a imagem é um dos recursos mais utilizados e de importância relevante no auxílio ao diagnóstico médico.

O processamento de imagens médicas requer uma quantidade grande de recursos de *hardware* devido ao elevado custo computacional de processamento, tornando-se inviável para quem não dispõe de tais recursos, uma vez que estes requerem curto tempo de resposta e não se permite armazenamento com perdas de dados.

O tamanho da imagem elevado aliado à necessidade de aplicação de algum filtro, seja para suavização, atenuação ou realce, aumenta o tempo de processamento dessas imagens, prejudicando a avaliação das mesmas [6].

Uma solução encontrada para suprir a necessidade de alto poder computacional para o processamento desse tipo de imagem é a utilização de sistemas distribuídos e de bibliotecas de passagem de mensagem, que viabilizam a computação paralela sobre sistemas distribuídos (computação paralela distribuída).

Este artigo apresenta a implementação de algoritmos de suavização e detecção de bordas de imagens de forma seqüencial e paralela aplicados às imagens mamográficas.

2. Computação Paralela Distribuída

Os sistemas baseados em *cluster* de computadores apresentam a melhor relação custo/desempenho para a execução de um amplo número de aplicações de alto desempenho, incluindo o processamento de imagem [4].

A programação paralela com base na troca de mensagem requer do programador controle sobre a distribuição e troca de dados, além da especificação explícita da execução paralela do código entre os diferentes processadores, o que impõe um alto grau de dificuldade quando comparada à programação seqüencial [8]. Os sistemas baseados na troca de mensagens mais utilizados são o MPI (*Message Passing Interface*) e o PVM (*Parallel Virtual Machine*).

Existem diversas implementações de MPI e PVM em aplicações desenvolvidas em linguagens como Fortran, C e C++. Com o surgimento de Java, inúmeras propostas foram apresentadas para a utilização dessas bibliotecas nessa linguagem as quais se pode citar o mpiJava e o JPVM (*Java Parallel Virtual Machine*) [7], sendo este último o ambiente de passagem de mensagem utilizado neste artigo.

2.1 JPVM

JPVM é uma API implementada totalmente em Java o que permite a troca de mensagens explícitas, baseado pela biblioteca PVM, fornecendo uma interface semelhante, com o intuito de facilitar para o programador acostumado com o PVM.

Devido à característica da linguagem Java o JPVM apresenta um nível de portabilidade, interoperabilidade superior e algumas características não achadas no PVM padrão, como segurança em *threads*. O JPVM é composto [2]:

- jpvmDaemon: um processo que é executado em todos os nós, formando uma máquina virtual paralela, realiza a comunicação entre os processos criados e coordena as tarefas em execução;

- jpvmEnvironment: biblioteca que trata funções básicas, como troca de mensagens, criação e eliminação de processos, sincronização de tarefas.

A comunicação é feita de forma direta tarefa-para-tarefa implementada sobre TCP *sockets* usando JAVA *objeto serialization interface* [2].

3. Modelo de Paralelismo em Processamento de Imagens

Os requisitos básicos de um sistema de processamento paralelo de imagem consistem em uma infra-estrutura composta essencialmente de funções de comunicação e distribuição de dados adequados que permita executar de forma eficiente quaisquer algoritmos de imagens [1].

Inicialmente foram escolhidas técnicas de suavização e detecção de bordas, a fim de verificar a relação custo/benefício decorrente da aplicação das tecnologias de computação paralela distribuída.

As técnicas de suavização são utilizadas em uma etapa de pré-processamento para a redução de ruídos e para a remoção de pequenos detalhes de uma imagem antes da extração de objetos, como também, para conexão de pequenas descontinuidades em linhas e curvas [3][6].

Uma técnica comum de suavização é o filtro de mediana que consiste em substituir o valor de um determinado *pixel* pelo valor mediano da sua vizinhança. O valor mediano é o valor central obtido quando se ordena os *pixels* da vizinhança.

A detecção de bordas é outro exemplo de algoritmo que usa operações baseadas em vizinhança.

A detecção de bordas é útil principalmente quando se deseja conhecer informações a respeito de tamanho e forma dos objetos representados na imagem.

Para avaliação neste artigo foi utilizado o algoritmo de detecção de bordas fazendo uso dos operadores de Sobel [3].

O operador de Sobel calcula o valor absoluto aproximado do gradiente em cada ponto da imagem analisada, deixando em maior evidência as áreas cuja freqüência espacial possui um valor alto e que correspondem as bordas da imagem [3].

Uma proposta de paralelização eficiente é a divisão da imagem em blocos distribuídos pelos processadores, de forma a processar ao mesmo tempo vários blocos de uma mesma imagem.

Como pode ser observado na Figura 1, o nó mestre divide e distribui os dados para os outros nós. Cada nó processa sua parte recebida da imagem e então o envia ao mestre que recompõe a imagem [5]. Os algoritmos abordados neste artigo fazem uso de máscaras coeficientes que operam sobre uma "vizinhança" de pontos da imagem, sendo necessária alguma redundância nos blocos para o processamento [7].

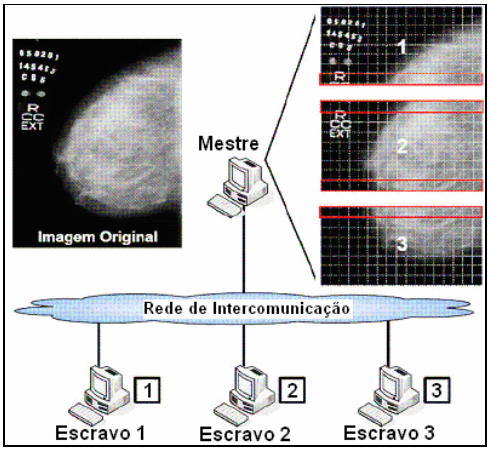

Figura 1 - Estratégia de paralelização de imagens médicas [7]

O desafio a ser vencido, mais especificamente em imagens médicas, é a divisão da imagem em blocos e a posterior junção desses blocos sem perdas de dados e de processamento.

4. Análise de Desempenho dos Resultados Obtidos

A análise de desempenho dos algoritmos de filtro de mediana e de detecção de bordas foi realizada através de diferentes testes reais em um ambiente paralelo distribuído controlado, composto por 9 máquinas homogêneas (Pentium IV de 2.7GHz com 512Mbytes de RAM, interligadas por uma rede *ethernet* de 100Mb/s).

Foram utilizados dois tipos de imagens mamográficas no formato TIFF com resolução de 16 *bits*, as imagens possuem tamanho elevado, primeira de aproximadamente 11 *Megabytes* e a segunda de aproximadamente 21 *Megabytes.*

O filtro de mediana foi avaliado com *templates* de tamanhos diferentes (5x5 e 7x7) utilizando o algoritmo de ordenação *shellsort*. O filtro de detecção de bordas foi executado com *template* de tamanhos 9X9 e 11x11. Observa-se que o tamanho do *template* é o que define o tamanho da vizinhança a ser considerada. Dependendo do tipo da imagem, uma maior vizinhança indica um melhor resultado de processamento.

Após a realização dos testes obteve-se uma média dos tempos de processamento tanto para aplicação seqüencial quanto para a paralela para os diferentes tipos de máscaras, o que permitiu efetuar uma análise estatística dos resultados obtidos.

Nas Figuras 2 e 3 são apresentados os resultados levando em consideração a máscara 5x5 e 7x7 respectivamente em imagens de 11Mb. Pelas figuras é possível observar que o tempo de execução do algoritmo seqüencial a partir do uso dessas máscaras é significativamente pior que o tempo de execução em paralelo. Isso pode ser observado principalmente para situações onde o número de processos equivale ou é maior que o número de máquinas.

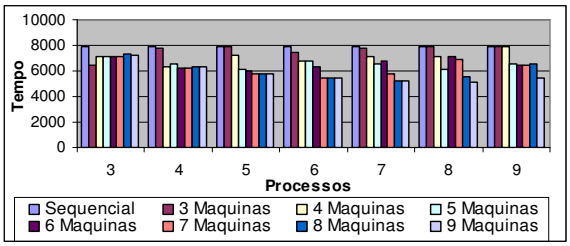

Figura 2 - Média do tempo de execução do filtro de mediana de máscara 5x5, imagem de 11MB.

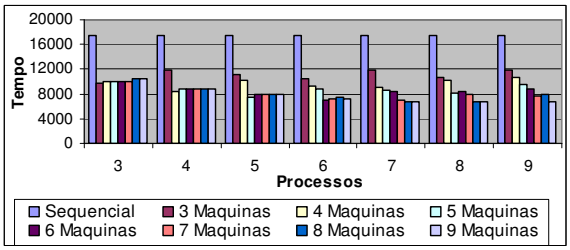

Figura 3 - Média do tempo de execução do filtro de mediana de máscara 7x7, imagem de 11MB.

Nas Figuras 4 e 5 são apresentados os resultados da execução seqüencial do filtro de detecção de bordas bem como as execuções em paralelo do mesmo algoritmo em 9 máquinas utilizando máscara de tamanho 9x9 e 11x11 em imagens de 11Mb.

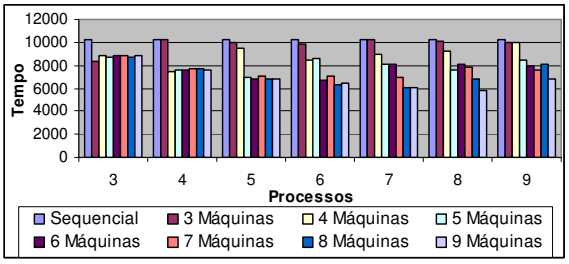

Figura 4 – Média do tempo de execução do filtro de detecção de borda de máscara 9x9, imagem de 11MB.

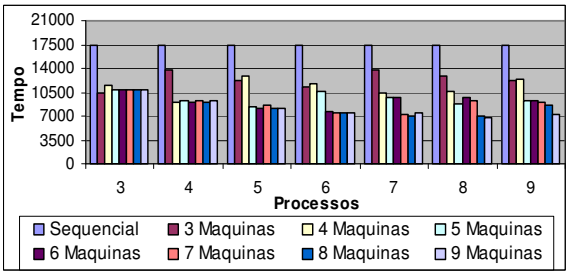

Figura 5 - Média do tempo de execução do filtro de detecção de borda de máscara 11x11, imagem de 11MB.

São apresentados nas Figuras 6 e 7 os resultados levando em consideração a máscara 5x5 e 7x7 respectivamente em imagens de 21Mb.

Tanto na Figura 6 como na 7 é pode-se observar significativa melhora de desempenho quando se faz uso de uma arquitetura paralela distribuída. Isso se dá pelo fato dos cálculos efetuados pelas máscaras serem volumosos, garantindo uma melhoria do uso em paralelo uma vez que a comunicação imposta se torna desprezível quando comparada ao ganho em relação aos cálculos efetuados.

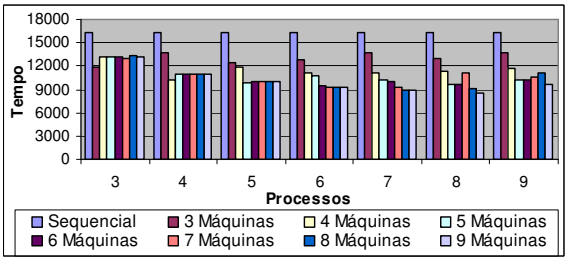

Figura 6 - Média do tempo de execução do filtro de mediana de máscara 5x5, imagem de 21MB.

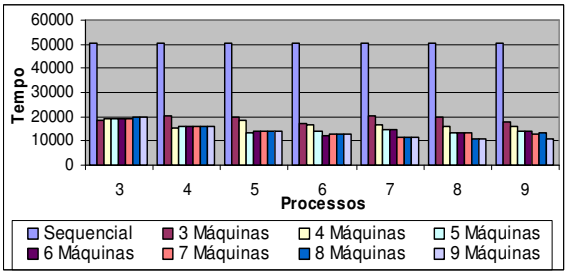

Figura 7 - Média do tempo de execução do filtro de mediana de máscara 7x7, imagem de 21MB.

Nas Figuras 8 e 9 são apresentados os resultados da execução seqüencial do filtro de detecção de bordas bem como as execuções em paralelo do mesmo algoritmo em 9 máquinas utilizando máscara de tamanho 9x9 e 11x11 em imagens de 21Mb. Mais uma vez pode-se observar o ganho em se fazer uso da

computação paralela distribuída, principalmente quando ocorre um aumento considerável no tamanho da imagem.

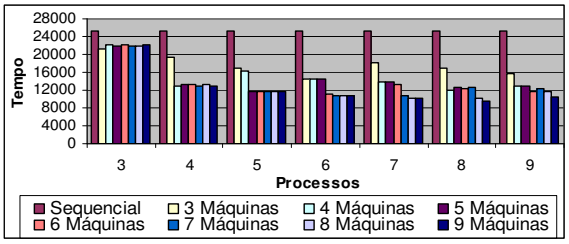

Figura 8 - Média do tempo de execução do filtro de detecção de borda de máscara 9x9, imagem de 21MB.

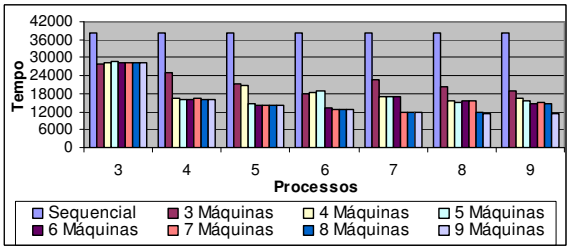

Figura 9 - Média do tempo de execução do filtro de detecção de borda de máscara 11x11, imagem de 21MB.

O número de processos mostrou ser um fator muito importante. No caso da biblioteca JPVM, na maioria dos testes o melhor desempenho ocorreu quando o número de processos foi o mesmo que o número de máquinas que forma a máquina virtual.

Pode-se observar também que o uso computação paralela distribuída em imagens de tamanho mais elevado agrega um ganho ainda maior de desempenho. Com imagens de 11 Mb foi possível obter ganhos de até 60% sobre a imagem seqüencial, enquanto que em imagens de 21 Mb foi possível obter ganhos de até 80% sobre a execução seqüencial.

5. Conclusões

Partindo-se dos resultados obtidos pode-se verificar que existe um ganho em se fazer uso do processamento paralelo distribuído quando se pensa em processamento de imagens médicas. Acredita-se que a utilização de outros filtros no domínio espacial também possa prover melhora significativa de desempenho para a versão paralela quando comparada à execução seqüencial.

Pode-se observar que o gargalo encontra-se no envio das mensagens geradas pelo JPVM, visto que a biblioteca JPVM utiliza o protocolo TCP para comunicação. Este gargalo é aceitável, pois se tratando de imagens médicas é necessária uma comunicação confiável para que não ocorra perda de dados.

Com base nesses resultados obtidos têm-se como trabalhos futuros o desenvolvimento de novos algoritmos de processamento de imagens. Além disso, testes adicionais devem ser executados o que permitirá a obtenção de um conjunto maior de dados a partir do qual tais informações poderão ser extraídas com maior fidelidade a fim de construir uma base de comparação efetiva.

6. Referências

[1] Barbosa, Jorge M. G. Paralelismo em Processamento e Análise de Imagem Médica. Tese apresentada ao Departamento de Engenharia Electrotécnica e de Computadores – Faculdade de Engenharia da Universidade do Porto, 2000.

[2] Ferrari, A. J.; *JPVM: Network parallel computing in Java*, In ACM Workshop on Java for High-Performance Network Computing, Palo Alto, 1998.

[3] Gonzalez, R. C.; Woods, R. E. Processamento de Imagens Digitais. Ed Edgard Blücher, São Paulo, 2002.

[4] Handi M.; Lee C. K., *Adaptive load-balancing of image processing applications on clusters of workstations*, IEEE Trans. on Computers, vol. 39, no.10, pp 1477-1232 1997.

[5] Nicolescu C.; Jonker P.P., *A data and task parallel image processing environment*, Parallel Computing, vol. 28, no. 7-8, pp 945-965, 2002.

[6] Nunes, F. L. S. M. Investigações em Processamento de Imagens Mamográficas para Auxílio ao Diagnóstico de Mamas Densas. Tese apresentada ao Instituto de Física de São Carlos – Universidade de São Paulo, 2001.

[7] Sabatine, R. J.; Saito P. T. M.; Nunes, F. L. S. M.; Branco, K. R. L. J. C. Uso da Computação Paralela Distribuída para Melhoria no Tempo de Processamento de Imagens Médicas. XIV Escola Regional de Informática da SBC - ERI-PR, pp 36-47,2007.

[8] Seinstra, F.J.; Koelma, D. *User Transparency: A Fully Sequential Programming Model for Efficient Data Parallel Image Processing*, Concurrency and Computation: Practice and Experience, 2004.

Agradecimentos

Os autores gostariam de agradecer à agência de financiamento FAPESP pelo apoio dado aos projetos do LAS da UNIVEM, principalmente pelos processos nº 2006/06671-0 e nº 2007/00868-0.