MODELO DE PARALELISMO PARA PROCESSAMENTO DE IMAGENS MÉDICAS[1]

Priscila Tiemi Maeda Saito[2]

Resumo

Em geral, processamento de imagens exige um alto poder de processamento. Quando se trata de imagens médicas, esta questão é ainda mais realçada, visto que esta classe de imagens não pode permitir armazenamento com perdas de dados e, muitas vezes, exige precisão na sua aquisição. Essa precisão gera um conjunto ainda maior de dados, o que prejudica a avaliação da mesma, dada a demora em se efetuar a passagem de algum filtro, seja para suavização, atenuação ou outro. A computação paralela e distribuída, viabilizada por sistemas distribuídos e bibliotecas de passagem de mensagens, pode oferecer a potência computacional adequada a este tipo de aplicação. Nesse projeto são implementadas técnicas de processamento de imagens de forma seqüencial e paralela, utilizando a linguagem de programação Java juntamente com a API JAI (Java Advanced Imaging) e a biblioteca de troca de mensagens mpiJava. Após essa implementação, é construída uma base de comparação entre a aplicação seqüencial e a paralela, a fim de avaliar o ganho com o paralelismo. Pretende-se, assim, obter um estudo que defina requisitos e subsídios para a aplicação da computação paralela e distribuída no processamento de imagens médicas.

Palavras-chave: Computação Paralela. Processamento de Imagens. Sistemas Distribuídos. Imagens Médicas. mpiJava.

Abstract

Image processing demands a high capacity processing. In medical images, this question is still more enhanced, since this class of images can not allow storage with losses of data and, many times, still demands precision in its acquisition. This situation can generate a bigger set of data, which harms the evaluation of the same one, given the delay in effecting the passing of some filter, either for suavization, attenuation or another one. The parallel and distributed computing, made possible by distributed systems and message passing libraries, it can offer the adequate computational power to this application. In this project, images processing techniques are implemented in sequential and parallel form, using the Java programming language together with API JAI (Java Advanced Imaging) and the mpiJava message passing library. After this implementation, a comparison base is formed between the sequential and parallel aplication to evaluate the profit with the parallelism. It is intended to get a study that defines requirements and subsidies for the application of the parallel and distributed computing in the medical images processing.

Keywords: Parallel Computing. Image Processing. Distributed Systems. Medical Images. mpiJava.

[1] Trabalho orientado pela professora Kalinka Regina L. Jaquie Castelo Branco do Instituto de Ciências Matemáticas e de Computação da USP.

[2] Bacharel em Ciência da Computação pelo UNIVEM.

INTRODUÇÃO

Imagens médicas têm por finalidade o auxílio na composição do diagnóstico de anomalias e o fornecimento de material para acompanhamento de terapias.

Sistemas de diagnóstico auxiliado por computador (computer-aided diagnosis - CAD) podem auxiliar o radiologista na tomada de decisão a respeito de um diagnóstico obtido por resultados de uma análise computadorizada de imagens médicas (GIGER, 2000).

Tais sistemas são objetos de pesquisa de várias instituições, onde vários autores destacam sua importância, apresentando taxas de diagnósticos errados em programas de rastreamento e mostrando que a utilização de esquemas CAD pode melhorar o desempenho de radiologistas no diagnóstico médico.

No entanto, sistemas deste tipo, aplicados no dia-a-dia da prática médica, são poucos, devido à necessidade de alto desempenho exigido por esta classe de sistema, tanto em nível de velocidade de execução quanto em nível de acerto nos resultados (NUNES, 2006). Isso acontece, pois alguns tipos de erros não são admitidos, visto que esses resultados são utilizados na tomada de decisão, seja para diagnósticos ou escolha de tratamento.

O desenvolvimento de sistemas de auxílio ao diagnóstico precisa vencer muitos desafios. Uma vez que as imagens médicas apresentam grande volume de dados para processamento e este processamento envolve, geralmente, milhares de operações em tempo real, é necessária a utilização de recursos computacionais que proporcionem alto desempenho.

Por esses e outros motivos, ainda não foi divulgada a aplicação de sistemas de auxílio ao diagnóstico brasileiros em nível clínico (NUNES, 2006).

A aplicação de técnicas de computação paralela e distribuída é uma linha de pesquisa que pode ser promissora no sentido de auxiliar na minimização desta questão. A seção seguinte apresenta o ambiente de passagem de mensagens como suporte para a computação paralela distribuída.

I SUPORTE À COMPUTAÇÃO PARALELA DISTRIBUÍDA

Os sistemas computacionais distribuídos aplicados à computação paralela permitem uma melhor relação custo/benefício para a computação paralela. Estes sistemas oferecem a potência computacional adequada às aplicações que não necessitam de uma máquina maciçamente paralela, porém necessitam de uma potência computacional maior que uma máquina seqüencial pode oferecer (BRANCO, 1999).

A computação paralela sobre sistemas distribuídos exige uma camada de software que possa gerenciar o uso paralelo, pois existe a necessidade da passagem de informações entre as várias máquinas que compõem o ambiente.

Ambientes de passagem de mensagens (ou interfaces de passagem de mensagens) são bibliotecas de comunicação que atuam como extensões de linguagens seqüenciais, como C e Fortran e provêem recursos necessários à programação paralela, como criação, comunicação e sincronização de processos, possibilitando, dessa forma, a construção de aplicações paralelas (SANTANA, 1997).

O MPI (Message Passing Interface) (SNIR et al., 1996), proposto para ser um padrão internacional de biblioteca de passagem de mensagens, tem sido constantemente aperfeiçoado, obtendo destaque na literatura, não só pela flexibilidade, mas também pelo fato de constituir um tipo de solução para o problema da portabilidade de programas paralelos entre sistemas diferentes. Além disso, possibilita eficiência e segurança em qualquer plataforma (CÁCERES, 2001), bem como o desenvolvimento de aplicações paralelas a um custo relativamente baixo em relação às máquinas paralelas (BEGUELIN et al., 1994) (SANTANA, 1997).

Existem muitas implementações de MPI em aplicações desenvolvidas em linguagens como Fortran, C e C++. Com o surgimento de Java (SUN, 2006), inúmeras propostas foram apresentadas para a utilização dessa biblioteca nessa linguagem, na qual se pode citar o mpiJava (BAKER et al., 1998) (MPIJAVA, 2007),

ambiente de passagem de mensagens utilizado neste trabalho.

1.1 mpiJava

O mpiJava é uma interface, amplamente utilizada na computação paralela e distribuída, que permite fazer uso da orientação a objetos em Java juntamente com a biblioteca MPI (BAKER et al., 1998).

Dentro das formas de comunicação possíveis em Java, a biblioteca apresenta bons resultados (BAKER et al., 1999) (SAITO et al., 2007e; SAITO et al., 2007f). O uso de mpiJava já foi validado em diversas implementações e avaliações de desempenho (TABOADA et al., 2003).

Além disso, a escolha desse ambiente para o processamento paralelo dos algoritmos vem em decorrência do uso da linguagem de programação Java que contém fatores como: portabilidade (permitindo a independência de plataforma), simplicidade, clareza nos códigos, funcionalidades (auxiliando no desenvolvimento de aplicações paralelas), bem como a existência de APIs especializadas que possibilitam o uso, cada vez maior, desta linguagem em processamento de imagens (exemplo, utilização da API JAI nesse trabalho).

2 PROCESSAMENTO DE IMAGENS MÉDICAS

A finalidade das imagens médicas é auxiliar na composição do diagnóstico de anomalias e fornecer material para acompanhamento de terapias, sendo estas imagens provenientes de diversos tipos de modalidades como Radiografia, Ultra-Sonografia (USa), Ressonância Magnética Nuclear (RMN), Tomografia Computadorizada (TC), entre outras (NUNES, 2006).

As técnicas de processamento de imagem, reconhecimento de padrões, entre outras, são aplicadas com o objetivo de melhorar as imagens e extrair delas informações úteis aos diagnósticos.

Existem inúmeros métodos de processamento de imagens. A escolha de procedimentos a serem aplicados depende do objetivo que se deseja em relação a uma determinada categoria de imagem. Somente após esta definição, é possível traçar estratégias a partir da utilização de uma técnica de processamento, da combinação de várias delas ou, ainda, da criação de novas técnicas.

De forma geral, as operações com imagens podem ser classificadas em baixo nível (pré-processamento), nível médio (segmentação) e nível alto (reconhecimento de padrões).

Neste trabalho, o objetivo é verificar o desempenho de técnicas de suavização (filtro de mediana) e segmentação (filtro de detecção de bordas) implementadas de forma seqüencial e paralela, a fim de verificar a relação custo-benefício decorrente da aplicação das tecnologias de computação paralela distribuída.

2.1 Suavização

Filtros de suavização são utilizados 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 (GONZALEZ, 2002). Entre as técnicas mais comuns de suavização estão os filtros de média e mediana.

A filtragem mediana usada neste trabalho (GONZALEZ, 2002) consiste em substituir o valor de um determinado pixel pelo valor mediano da sua vizinhança que é o valor central obtido quando se ordena os pixels da vizinhança.

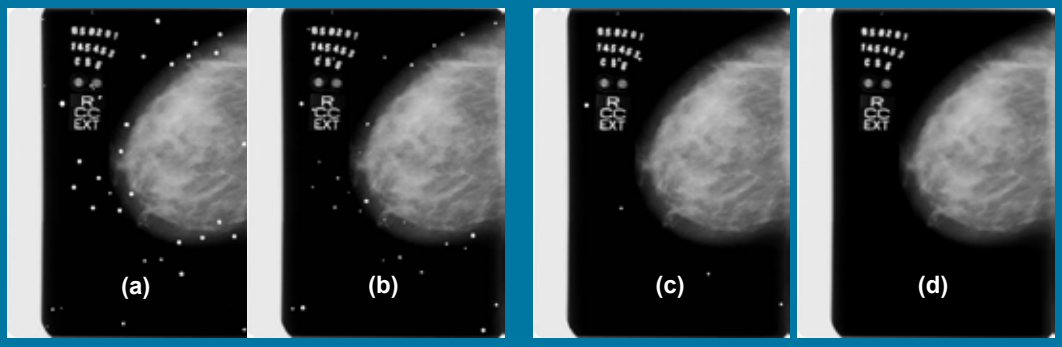
Na Figura 1, são apresentados exemplos da aplicação da filtragem mediana em imagens médicas utilizadas para a análise dos testes de desempenho.

Observa-se que a quantidade de ruído presente na imagem original, Figura 1(a), é grande. Com a aplicação do filtro, pode-se observar pelas Figuras 1(b), (c) e (d) uma diminuição desse ruído, de modo que, quanto maior o tamanho da máscara utilizada, mais suavizada será a imagem resultante e, conseqüentemente, menos ruído essa imagem apresentará.

2.2 Detecção de Bordas

A detecção de bordas é outro exemplo de algoritmo que usa operações ba-

Figura 1 - Exemplos de suavização utilizando o filtro mediana. (a) imagem mamográfica original; (b) imagem suavizada com máscara 3x3; (c) imagem suavizada com máscara 5x5; (d) imagem suavizada com máscara 7x7.



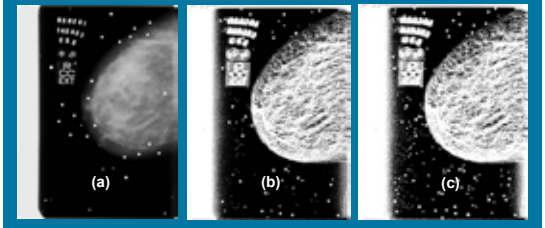
seadas em vizinhança. Representar uma imagem por meio de suas bordas pode ser vantajoso para muitos esquemas na área médica, visto que as bordas fornecem informações úteis para a composição de diagnósticos.

Para avaliação neste trabalho foi utilizado o algoritmo de detecção de bordas, baseando-se nos operadores de Sobel (GONZALEZ, 2002), porém algumas modificações foram realizadas. Estes operadores calculam 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 em relação à vizinhança e que correspondem às bordas da imagem.

A Figura 2 apresenta exemplos do resultado obtido com a aplicação desses operadores.

Na Figura 2(a), observa-se que a quantidade de ruído presente na imagem original é grande. Com a aplicação do filtro, pode-se observar, pelas Figuras 2(b) e (c), um realce em suas bordas. Sendo que, quanto maior o tamanho da máscara utilizada, mais realçada será a imagem resultante, porém realça-se também o ruído presente na imagem original.

Figura 2 - Exemplos de detecção de bordas utilizando os operadores (modificados) de Sobel. (a) imagem mamográfica original; (b) imagem com máscara 9x9; (c) imagem com máscara 11x11.



3 APLICAÇÃO DA COMPUTAÇÃO PARALELA NA OTIMIZAÇÃO DO PROCESSAMENTO DE IMAGENS MÉDICAS

O tempo de processamento computacional de uma imagem deve ser observado, principalmente, quando são desenvolvidos sistemas com o objetivo de execução em tempo real.

O processamento no domínio espacial geralmente percorre a imagem e altera, de alguma forma, o valor de cada pixel. Quando uma imagem é grande, devido, principalmente, à sua resolução espacial, deve-se aperfeiçoar o processamento a fim de torná-lo viável em tempo real.

É necessário o estabelecimento de algoritmos eficazes que possam fornecer, no menor tempo possível, um desempenho satisfatório em termos de acerto, a fim de contribuir, de fato, para o auxílio ao diagnóstico.

3.1 Desenvolvimento do Programa em Paralelo

O requisito básico de um sistema de processamento paralelo de imagens consiste em uma infra-estrutura que permita a execução eficiente de algoritmos

neste domínio, sejam de nível baixo - que executam alterações globais na imagem; de nível médio - que identificam estruturas importantes na imagem; ou de nível alto - relacionados com o reconhecimento de padrões. Essa infra-estrutura é composta essencialmente de funções de comunicação e distribuição de dados adequados ao processamento de imagens (BARBOSA, 2000).

Como mencionado anteriormente, os filtros executados no domínio espacial manipulam diretamente os pixels que compõem a imagem. São os mais utilizados devido à facilidade de implementação, mas exigem alto poder de processamento, visto que as imagens constituem, na maioria das vezes, matrizes enormes de pontos a serem processados. É neste contexto que as técnicas de processamento de imagens, em especial aquelas desenvolvidas especificamente para aplicação em imagens médicas, podem se beneficiar dos conceitos de paralelização de imagens.

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

Definir o tipo de paralelismo que melhor se adapte ao processamento proposto é tarefa que permite obtenção de melhor desempenho. Desse modo, o paralelismo de dados é o que melhor se enquadra neste caso, pois pode-se definir que o processador execute as mesmas tarefas sobre diferentes dados aproximadamente do mesmo tamanho, tendo um único fluxo de controle (SPMD - Single Process Multiple Data).

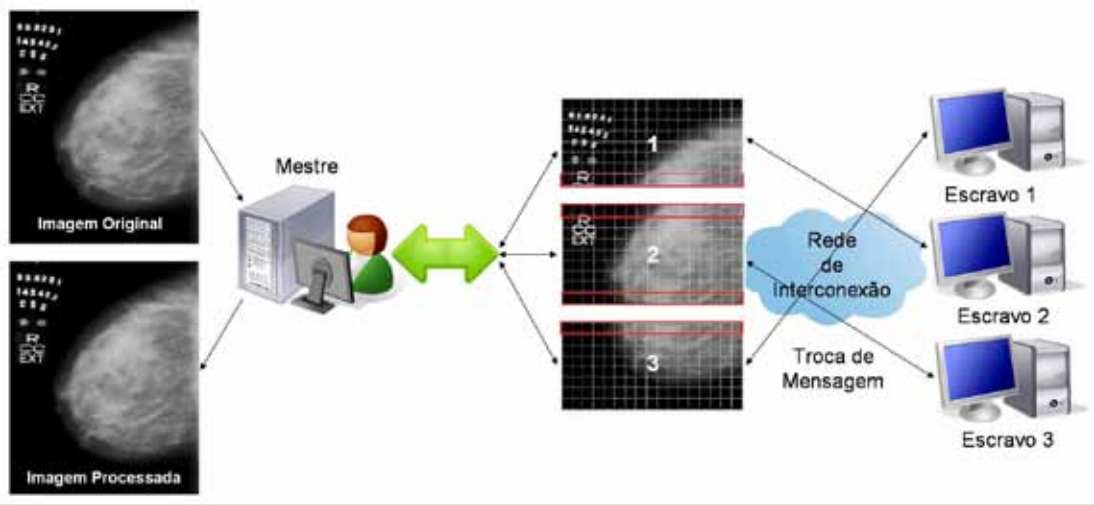
O desafio a ser vencido, neste caso e mais especificamente em imagens médicas, é a divisão da imagem em blocos e a posterior junção desses blocos sem perdas de processamento.

A Figura 3 ilustra uma estratégia de paralelização [SAITO et al., 2007g] em que uma imagem médica é dividida em blocos pelo mestre. Os blocos são subseqüentemente transmitidos por meio da biblioteca mpiJava aos escravos. Estes têm a incumbência de processá-los e devolvê-los já processados ao mestre, o qual os une, reconstituindo a imagem.

Pode-se perceber também que os blocos apresentam tamanhos variados (como observado pelo número de linhas em cada um dos blocos 1, 2 e 3 apresentados na Figura 3), dependendo da característica de cada algoritmo. Os algoritmos dos filtros mediana e de detecção de bordas, abordados neste trabalho, fazem uso de máscaras coeficientes que operam sobre uma "vizinhança" de pontos da imagem. Sendo assim, há necessidade de alguma redundância nos blocos para o processamento (destacada em vermelho na Figura 3) e que deve ser retirada na reconstituição da imagem.

Um algoritmo paralelo genérico, em

Figura 3. Estratégia de paralelização para o processamento das imagens médicas (SAITO et al., 2007g).



que o mestre efetua a distribuição aos escravos, é apresentado na Figura 4. Os detalhes da implementação dos algoritmos paralelos são discutidos em [SAITO, 2007h].

4 ANÁLISE DE DESEMPENHO

Figura 4. Implementação genérica do algoritmo paralelo utilizado para o processamento de imagens.

```
abre a imagem
define num_processos
caso seja mestre
  para i = 0 até num_processos
    tam_bloco = (altura_imagem
div num_processos)+máscara
    define bloco com tam_bloco
    para j = 0 até tam_bloco
      copia pixels da imagem no
bloco
  envia(bloco)
  para k = 0 até num_processos
    recebe(bloco)
    para j = 0 até tam_bloco
      copia pixels do bloco na
imagem
caso seja escravo
  recebe(bloco)
  processa(bloco)
  envia(bloco)
```

DOS RESULTADOS OBTIDOS

A análise de desempenho dos algoritmos de suavização (filtro mediana) e de detecção de bordas (filtro de Sobel) foi realizada por meio de diferentes testes reais em um ambiente paralelo distribuído composto inicialmente por 3 máquinas homogêneas (Pentium IV de 2.7GHz com 512Mbytes, interligadas por uma rede ethernet de 100Mb/s), sendo, posteriormente, acrescidas máquinas idênticas para compor um ambiente de até 10 máquinas.

Foram utilizadas imagens mamográficas (radiografia das mamas) de diferentes tamanhos (500 KB, 1 MB, 11 MB e 21 MB), no formato TIFF, com resolução de contraste de 16 bits, consistindo em matrizes de tamanho médio 432x580 pixels, 625x840 pixels, 2500x3000 pixels e 2855x3835 pixels, respectivamente. Tais imagens fazem parte de um banco de imagens desenvolvido pelo LAPIMO (Laboratório de Processamento de Imagens Médicas e Odontológicas, da EESC/USP).

O filtro mediana foi avaliado com máscaras de tamanhos diferentes 3x3, 5x5 e 7x7, utilizando o algoritmo de ordenação shellsort. O filtro de detecção de bordas foi avaliado com máscaras de tamanhos 9x9 e 11x11. Observa-se que o tamanho da máscara é o que define o tamanho da vizinhança a ser considerada e, dependendo do tipo da imagem, uma maior vizinhança pode levar a um melhor resultado de processamento paralelo.

Após a realização dos testes, obteve-se uma média dos 30 tempos de processamento tanto para aplicação seqüencial quanto para a paralela para os diferentes tipos de máscaras e tamanho de imagens.

Quando da avaliação da execução em paralelo, foram efetuados testes com até dez máquinas, e para cada uma delas, testes com até 11 processos sendo iniciados em paralelo, sendo possível, assim, realizar a comparação de desempenho de cada uma das possíveis combinações, por meio das medidas relacionadas ao tempo (em milissegundos), speedup e eficiência. Além disso, todos os resultados obtidos foram avaliados estatisticamente, por meio de testes de hipóteses (JAIN, 1991), para comprovar seu grau de significância.

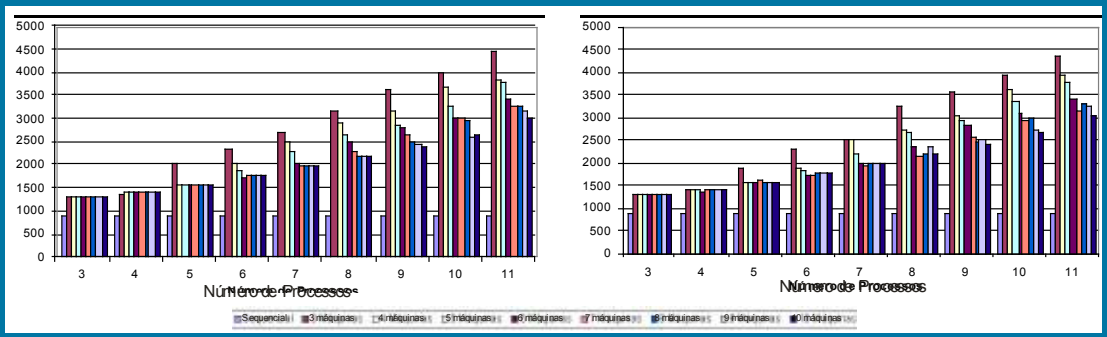
As próximas seções apresentam os resultados da execução seqüencial e paralela das técnicas implementadas utilizando-se imagens de diferentes tamanhos.

4.1 Imagens de Tamanhos Pequenos - 500 KB e 1 MB

Nas análises de desempenho das implementações dos filtros mediana e de detecção de bordas Sobel para imagens de tamanhos pequenos (500 KB e 1 MB), foi possível observar que o tempo de execução do algoritmo seqüencial é significativamente melhor que o tempo do mesmo em paralelo independentemente do tamanho da máscara, do número de máquinas e da quantidade de processos iniciados em paralelo.

As Figuras 5 e 6 ilustram, respectivamente, o resultado da execução seqüencial e paralela em imagens de 500 KB e 1 MB,

Figura 5. Gráfico da execução da aplicação seqüencial comparada com a aplicação paralela de imagens de 500KB, utilizando: (a) filtro mediana com máscara 7x7 e (b) filtro Sobel com máscara 11x11.



dos filtros mediana, com a utilização de máscara de tamanho 7x7 (Figuras 5(a) e 6(a)) e Sobel, com a utilização de máscara de tamanho 11x11 (Figuras 5(b) e 6(b)).

Uma vez que, tanto a quantidade de cálculos efetuados pelos filtros quanto os tamanhos das imagens utilizadas são relativamente pequenos, a paralelização dos mesmos não impõe melhoria, pois se consome mais tempo com comunicação para envio de dados, por meio da rede, do que no cálculo da máscara propriamente dita. Ou seja, o tempo utilizado para o envio de cada parte da imagem (subvetor) é maior que o tempo gasto pelos escravos para realizar o processamento (inúmeras multiplicações necessárias de acordo com o algoritmo), o que a torna uma aplicação mais voltada para comunicação do que para o processamento.

4.2 Imagens de Tamanho Médio – 11 MB

Foi possível observar que, com a utilização de imagens de 11 MB, da mesma forma com a utilização de imagens de tamanhos 500 KB e 1 MB, o tempo de execução do algoritmo seqüencial é ainda significativamente melhor que o tempo do mesmo em paralelo, independentemente do número de máquinas e da quantidade de processos iniciados em paralelo, utilizando-se máscaras de tamanho 3x3.

Para tamanhos de máscaras 5x5, em alguns casos, o tempo de execução do algoritmo paralelo passa a ser significativamente melhor que o tempo seqüencial. Isso pôde ser observado principalmente para situações em que o número de processos equivale ou é maior que o número de máquinas utilizadas.

À medida que se aumenta o número de processos para certa quantidade de máquinas, o desempenho diminui, tornando-se, em alguns casos, maior que o tempo médio seqüencial. Isso acontece porque quando, se aumenta o número de

Figura 6. Gráfico da execução da aplicação seqüencial comparada com a aplicação paralela de imagens de 1 MB, utilizando: (a) filtro mediana com máscara 7x7 e (b) filtro Sobel com máscara 11x11.

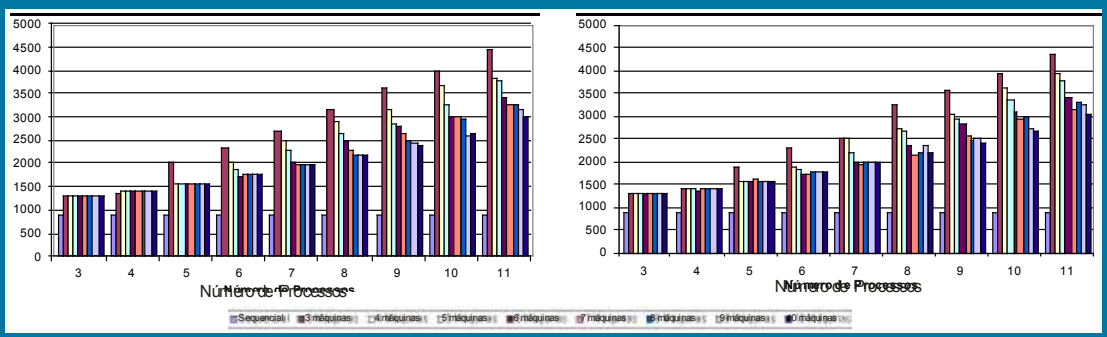
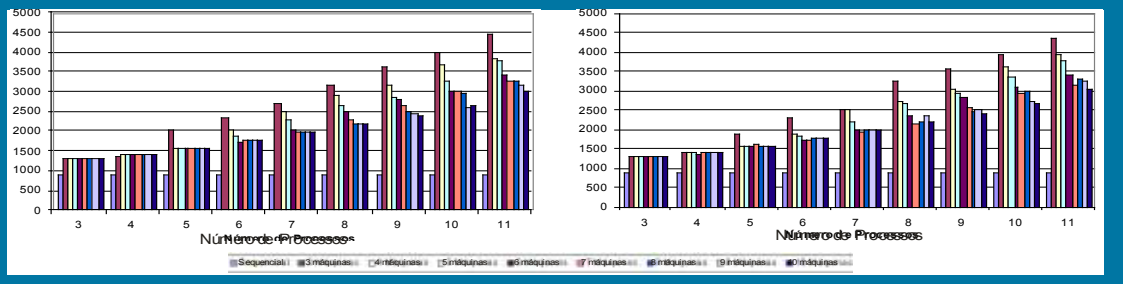


Figura 7. Gráfico da execução da aplicação seqüencial comparada com a aplicação paralela de imagens de 11 MB, utilizando: (a) filtro mediana com máscara 7x7 e (b) filtro Sobel com máscara 11x11.



processos, pode haver uma sobrecarga na comunicação, passando a ser um fator evidente de queda de desempenho.

Diferentemente das máscaras 3x3 e 5x5, quando utilizadas as máscaras 7x7 (filtro mediana), 9x9 e 11x11 (filtro de detecção de bordas), independentemente do número de processos iniciados e do número de máquinas, o tempo médio paralelo é sempre melhor que o tempo seqüencial, uma vez que o processamento é alto e o aumento no número de processos ainda implica em ganho de desempenho sobre a sobrecarga de comunicação.

As Figuras 7(a) e 7(b) ilustram, respectivamente, o resultado da execução seqüencial e paralela em imagens de 11 MB dos filtros mediana, com a utilização de máscara de tamanho 7x7, e Sobel com a utilização de máscara de tamanho 11x11.

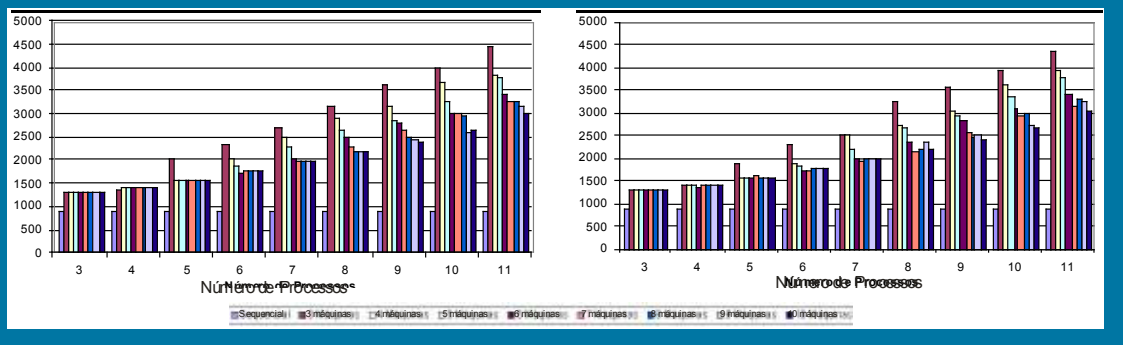
Essa diferença e significativa melhora de desempenho, quando se faz uso de uma arquitetura paralela distribuída, dá-se pelo fato dos cálculos efetuados pelas máscaras serem volumosos, o que garante uma melhoria do seu uso em paralelo, uma vez que a comunicação imposta se torna menos relevante quando comparada ao ganho em relação aos cálculos efetuados. Isso significa que a quantidade de comunicação exigida exerce menor influência na avaliação, visto que o volume de dados a serem transmitidos é menor que o volume de dados a ser processado.

O número de processos mostrou ser um fator muito importante. O melhor resultado é obtido quando se tem um número de processos maior (em uma unidade) que o número de máquinas participantes. Isso é devido ao mpiJava utilizar um processo mestre que é contabilizado, mas não efetua a tarefa escrava propriamente dita.

4.3 Imagens de Tamanho Grande – 21 MB

Para imagens de tamanho 21 MB, foi possível observar que o tempo de execução paralelo dos filtros, independentemente do número de máquinas utilizadas, do número de processos iniciados e do tamanho das máscaras (incluindo a 3x3) apresenta um melhor desempenho se comparado

Figura 8. Gráfico da execução da aplicação seqüencial comparada com a aplicação paralela de imagens de 21 MB, utilizando: (a) filtro mediana com máscara 7x7 e (b) filtro Sobel com máscara 11x11.



ao tempo seqüencial. A imagem utilizada é grande, dessa forma, os cálculos efetuados pelas máscaras são ainda mais volumosos, garantindo uma melhoria bastante significativa do uso em paralelo.

As Figuras 8(a) e 8(b) ilustram, respectivamente, o resultado da execução seqüencial e paralela em imagens de 21 MB dos filtros mediana, com a utilização de máscara de tamanho 7x7, e Sobel com a utilização de máscara de tamanho 11x11.

Como as técnicas apresentadas (filtro mediana e de detecção de bordas) trabalham no domínio espacial é interessante observar, ainda, que o tamanho da máscara exerce influência no desempenho do processamento, pois o esforço computacional aumenta proporcionalmente a esta dimensão devido ao incremento na quantidade de operações matemáticas que devem ser realizadas.

A seguir estão compiladas algumas considerações sobre avaliações de desempenho entre tamanhos de imagens e máscaras. Lembrando que foram utilizadas imagens de tamanhos 500 KB, 1 MB, 11 MB e 21 MB e máscaras de tamanhos 3x3, 5x5, 7x7, 9x9 e 11x11. Foram calculadas as medidas speedup e eficiência, e essas medidas foram representadas em percentuais, conforme pode ser observado pelas Tabelas 1 e 2.

Tabela 1. Eficiência (em %) na paralelização do filtro de mediana.

Tamanho	Imagem			
Máscara	500 KB	1 MB	11 MB	21 MB
3x3	-686,66	-452,76	-193,51	-187,01
5x5	-244,48	-179,57	80,74	74,31
7x7	-146,86	-100,58	45,89	42,03
11x11	-146,31	-107,87	31,92	43,32

Para a obtenção dos percentuais apresentados pelas Tabelas 1 e 2, foram calculados, para cada quantidade de máquinas, os speedups e eficiência dos seus melhores tempos de execução (em milissegundos). Obtida a eficiência de cada uma das quantidades de máquinas utilizadas, foi realizada uma média desses valores, representando os percentuais apresentados.

Tabela 2. Eficiência (em %) na paralelização do filtro de detecção de bordas.

Tamanho	Imagem			
Máscara	500 KB	1 MB	11 MB	21 MB
9x9	-205,40	-138,10	64,44	56,90

Exemplificando, para imagens de tamanho 21 MB, utilizando-se máscara de tamanho 7x7, obtiveram-se os melhores tempos de execução para cada quantidade de máquinas utilizadas. Para três máquinas, o melhor tempo médio (utilizando-se 4 processos) foi 19749,5 milissegundos. Os demais valores foram 16819,70; 16331,1; 15391; 15399,2; 15203,3; 15294,1 e 14931,6 para 4, 5, 6, 7, 8, 9 e 10 máquinas, respectivamente.

Foi calculado o speedup de cada um desses valores, obtendo-se, assim, uma média dos valores da eficiência encontrados por meio dos speedups. A média obtida foi de 0,75699025, determinando que a paralelização do filtro de detecção de bordas Sobel, utilizando-se imagem de tamanho 21 MB e máscara de tamanho 7x7, é 75,6990% mais eficiente que a aplicação do mesmo de forma seqüencial.

Pelas Tabelas 1 e 2, é possível perceber que o ganho de desempenho do tempo de processamento paralelo cresce em proporções muito grandes em relação ao tamanho da imagem utilizada. Esse crescimento pode ser melhor observado pelas curvas, referentes às máscaras 3x3, 5x5, 7x7, na Figura 9, e às máscaras 9x9 e 11x11, na Figura 10.

CONSIDERAÇÕES FINAIS

O presente trabalho de pesquisa teve como objetivo principal demonstrar a viabilidade na otimização do tempo de

Figura 9. Gráfico da eficiência (em %) na paralelização do filtro de mediana utilizando imagens de diferentes tamanhos.

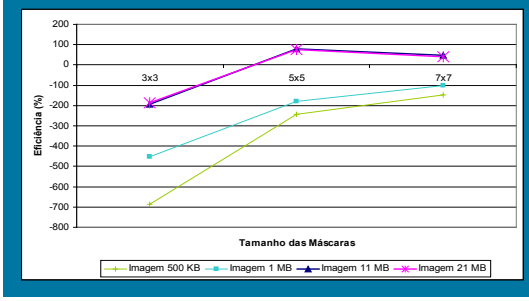
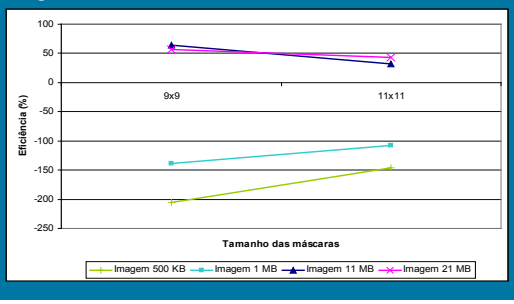


Figura 10. Gráfico da eficiência (em %) na paralelização do filtro de detecção de bordas utilizando imagens de diferentes tamanhos.



processamento de imagens, mais especificamente de imagens médicas. Para tanto, utilizou-se da programação paralela e distribuída, viabilizada por sistemas distribuídos e pela biblioteca de passagem de mensagens mpiJava.

Os objetivos, para algumas situações, foram alcançados e comprovados por meio de testes das implementações dos algoritmos estudados. 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.

Os algoritmos de suavização (filtro mediana) e de segmentação (detecção de bordas) foram estudados de forma detalhada para a obtenção de maior eficiência nas implementações realizadas. Após o domínio detalhado dos algoritmos, os mesmos foram implementados numa primeira instância na linguagem Java de forma seqüencial, em seguida, suas versões paralelas foram implementadas para realizar as avaliações de desempenho.

Nessa avaliação foram utilizados critérios que avaliaram tempo de execução, número de máquinas, número de processos, tamanho de máscaras e tamanho das imagens utilizadas.

Na análise de desempenho realizada por meio de testes das implementações dos algoritmos na forma seqüencial e paralela, foram obtidos os seguintes resultados e informações:

Independentemente do tipo de algoritmo de ordenação utilizado (bubblesort ou shellsort), na implementação do algoritmo filtro mediana, os tempos de execução em paralelo apresentaram-se satisfatórios, dependendo dos tamanhos das imagens e das máscaras utilizadas (SAITO et al., 2007c);

A média dos tempos de execução da implementação paralela do algoritmo utilizando o método shellsort é menor que a média dos tempos utilizando o método bubblesort, visto que a versão seqüencial deste último é muito mais lenta que a do shellsort (SAITO et al., 2007c);

Apesar do resultado não ser favorável à execução paralela dos filtros, utilizando-se imagens pequenas e, em alguns casos, na aplicação de máscaras de tamanho 3x3 e 5x5 (filtro mediana), foi possível observar que, para processamentos intensos, como é o caso da utilização de máscaras de tamanhos 7x7 (filtro mediana), 9x9 e 11x11 (filtro de detecção de bordas), o uso do processamento paralelo é bastante vantajoso (SAITO et al., 2007a, SAITO et al., 2007b, SAITO et al., 2007d);

A melhor média dos tempos de execuções é obtida quando se tem um processo a mais (em uma unidade) que o número de máquinas, conforme mencionado no capítulo anterior;

À medida que se aumenta o número de processos, diminui-se o desempenho, sendo um resultado já esperado, demonstrando o impacto que o tempo de comunicação entre as máquinas exerce sobre o tempo de processamento.

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. Com base nisso, o desenvolvimento de outros algoritmos de processamento de imagens, mais complexos, pode ser realizado em uma próxima etapa de projeto

Acredita-se também que a paralelização dos algoritmos de ordenação, utilizados na implementação do filtro mediana, possa prover melhora significativa no desempenho dos mesmos, uma vez que já se tem comprovado na literatura (CORTÉS, 1999) que a paralelização de algoritmos de ordenação permite um aumento de desempenho quando comparado a sua versão seqüencial.

Demais testes com outras modalidades de imagens, diferentes das mamo-

gráficas utilizadas até então, podem ser proferidos, permitindo, dessa forma, a obtenção de um conjunto maior de dados a partir do qual mais informações poderão ser extraídas.

Embora a computação paralela distribuída viabilize a transferência de dados e o uso de múltiplas máquinas, possibilitando os resultados obtidos e já apresentados, ela não provê por si própria (o ambiente de passagem de mensagem utilizado provê somente o escalonamento round-robin) mecanismos eficientes para a distribuição de processos a processadores (escalonamento de processos objetivando o balanceamento de cargas) (BRANCO, 2004).

O uso de escalonadores de processos ou até mesmo de mecanismos para prover uma distribuição voltada ao balanceamento de cargas deve prover uma melhora ainda mais significativa em relação aos resultados obtidos (FERRARI & ZHOU, 1987; BRANCO, 2004; BRANCO et al., 2006), e é nesse sentido que estudos devem ser efetuados para averiguar a qual classe de aplicação (CPU-Bound, Memory-Bound, Network-Bound ou Disk-Bound) as aplicações de processamento de imagens se enquadram e, posteriormente, averiguar quais índices de carga e/ou desempenho podem e devem ser utilizados para que ocorra uma melhora ainda maior no tempo de processamento desses algoritmos em paralelo.

AGRADECIMENTOS

As autoras gostariam de agradecer à agência de financiamento FAPESP pelo apoio dado aos projetos do Laboratório de Arquitetura de Sistemas (LAS) da UNIVEM, principalmente pelo processo nº 2006/06671-0.

REFERÊNCIAS

BAKER, M.; CARPENTER, B.; FOX, G.; KO, S. H.; LIM, S. **mpiJava:** An Object-Oriented Java Interface to MPI. Lecture notes in computer science. In: INTERNATIONAL WORKSHOP ON JAVA FOR PARALLEL AND DISTRIBUTED COMPUTING, IPPS/SPDP 1999, San Juan, Puerto Rico, 1999. Proceedings...Berlin, Alemanha: Springer, 1999. v. 1586, p. 748-762.

BAKER, M.; CARPENTER, B.; KO, S. H.; LI, X. (1998). **mpiJava:** A Java Interface to MPI. Lecture notes in computer science. In: UK Workshop on Java for High Performance Network Computing - EUROPAR, 1, 1998, Stanford. Proceedings... Stanfors: Springer – Verlag, 1998. p. 1-16.

BARBOSA, J. M. G. **Paralelismo em Processamento e Análise de Imagem Médica.** 2000, 240f. Tese (Doutorado) - Departamento de Engenharia Electrotécnica e de Computadores, Faculdade de Engenharia da Universidade do Porto, 2000.

BEGUELIN, A.; GEIST, A.; DONGARRA, J.; JIANG, W.; MANCHEK, R.; SUNDERAM, V. **PVM:** Parallel Virtual Machine. A User's Guide and Tutorial for NetWork Parallel Computing. Massachusetts Institute of Technology The MIT Press, s. 1, 1994.

BRANCO, K. R. L. J. C.; SANTANA, M. J.; SANTANA, R. H. C.; BRUSCHI, S. M. **PIV and WPIV:** Performance Index For Heterogeneous Systems Evaluation. In: 2006 IEEE INTERNATIONAL SYMPOSIUM ON INDUSTRIAL ELECTRONICS – ISIE'06, Montreal. Proceedings... Montreal, 2006. v. 1, p. 323-328.

BRANCO, K. R. L. J. C. **Índices de carga e desempenho em ambientes paralelos/distribuídos** – modelagem e métricas. São Paulo, 2004, 260 f. Tese (Doutorado em Ciência da Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação. Universidade de São Paulo, São Carlos, 2004.

BRANCO, K. R. L. J. C. **Extensão da Ferramenta de Apoio à Programação Paralela (F.A.P.P.) para Ambientes Paralelos Virtuais.** São Carlos, 1999, 152f. Dissertação (Mestrado em Ciência da Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação de São Carlos, Universidade de São Paulo, São Carlos, 1999.

CÁCERES, E. N.; MONGELLI, H.; SONG, S. W. **Algoritmos Paralelos Usando CGM/PVM/MPI:** Uma Introdução. In: AS TECNOLOGIAS da Informação e a Questão Social. Porto Alegre: Sociedade Brasileira de Computação, 2001.

CORTÉS, O. A. C. **Desenvolvimento e Avaliação de Algoritmos Numéricos Paralelos.** São Carlos, 1999. Dissertação (Mestrado em Ciência da Computação e Matemática Computacional) - Instituto de Ciências Matemáticas e de Computação - USP, São Carlos, São Paulo, 1999.

FERRARI, D.; ZHOU, S. **An Empirical Investigation of Load Indices for Load Balancing Applications.** In: PERFORMANCE'87 – INTERNATIONAL SYMPOSIUM ON COMPUTER PERFORMANCE MODELLING, MEASUREMENT AND EVALUATION. Proceedings... Amsterdam: Holland Publishing, 1987. p. 515-528.

GIGER, M. L. **Computer-Aided Diagnosis of Breast Lesions in Medical Images.** Computing in Science & Engineering, Los Alamitos, v. 2, n. 5, p. 39-45, 2000.

GONZALEZ, R. C.; WOODS, R. E. **Digital Image Processing.** 2 ed. Massachusetts: Addison-Wesley, 2002.

JAIN, R. **The Art of Computer Systems Performance Analysis:** Techniques for Experimental Design, Measurement, Simulation, and Modeling. New York: John Wiley & Sons, 1991.

MPIJAVA. **The HPJava Project document home page.** Disponível em: <http://www.hpjava.org/mpiJava/doc/api/>. Acesso em: jan. 2007.

NUNES, F. L. S. **Processamento de Imagens Médicas para Sistemas de Auxílio ao Diagnóstico.** In: RODELLO, I., A.; BREGA, J. R. F.; BRANCO, K. R. L. J. C. (Org). ERI – Escola Regional de Informática São Paulo / Oeste. São Paulo; Marília; Bauru: Estrela, 2006. cap. 4, p. 83-135.

SAITO, P. T. M.; SABATINE, R. J.; NUNES, F. L. S.; BRANCO, K. R. L. J. C. **Otimização de Algoritmos de Processamento de Imagens Médicas Utilizando a Computação Paralela.** Revista Diciplinarum Scientia, Santa Maria, v. 1, p. 1-11, 2007a.

_____. **Otimização de Algoritmos de Processamento de Imagens Médicas Utilizando a Computação Paralela.** In: SIMPÓSIO DE INFORMÁTICA DA REGIÃO CENTRO – SIRC 2007, 6, 2007, Santa Maria. Anais... Santa Maria: Centro Universitário Franciscano (UNIFRA), 2007b. p. 1-8

_____. **Otimização do Processamento de Imagens Médicas:** Uma Abordagem Utilizando Java. In: WORKSHOP DE VISÃO COMPUTACIONAL – WVC' 2007, 3, 2007, São José do Rio Preto. Anais... São José do Rio Preto: [S.n.], 2007c. p. 106 - 111

_____. **Otimização do Processamento de Imagens Médicas Utilizando a Computação Paralela.** In: CISCI 2007: CONFERENCIA IBEROAMERICANA EM SISTEMAS, CIBERNÉTICA E INFORMÁTICA, 6, 2007, Orlando, Flórida. Proceedings... Orlando: [S.n.], 2007d. v. 1, p. 1-6

_____. **Processamento de Imagens Médicas: Otimização no Tempo de Execução Usando Computação Paralela Distribuída.** In: SIMPÓSIO DE INSTRUMENTAÇÃO E IMAGENS MÉDICAS, 3, 2007, São Carlos. Anais... São Carlos, [S.n], 2007e. p. 6-9

_____. **Uma Abordagem Utilizando mpiJava para a Paralelização de Algoritmos de Processamento de Imagens.** In: WSCAD – WORKSHOP EM SISTEMAS COMPUTACIONAIS DE ALTO DESEMPENHO, 8, 2007, Gramado - Rio Grande do Sul. Anais... Gramado: [S.l], 2007f. p. 1-4.

_____. **Uso da Computação Paralela Distribuída para Melhoria no Tempo de Processamento de Imagens Médicas.** In: ERI/PR – ESCOLA REGIONAL DE INFORMÁTICA DA SBC, 14, 2007, Guarapuava. Anais... Guarapuava: [S.n.], 2007g. v. 1, p. 36-47.

SAITO, P. T. M.. **Otimização do Pro-**

cessamento de Imagens Médicas Utilizando a Computação Paralela. Marília, 2007. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação)- Centro Universitário Eurípides de Marília - UNIVEM, 2007h.

SANTANA, R. H. C.; SANTANA, M. J.; SOUZA, M. A.; SOUZA, P. S. L; PIEKARSKI, A. E. T. **Computação Paralela.** São Carlos: Universidade de São Paulo, Departamento de Ciências de Computação e Estatística, 1997. Lecture Notes.

SNIR, M.; OTTO, S.; HUSS-LEDERMAN, S.; WALKER, D.; DONGARRA, J. **MPI:** The Complete Reference. 2.ed. [S.l.]: The MIT Press; Massachusetts Institute of Techmology, 1996.

SUN Microsystems. **JAVA document home page.** Disponível em: <http://java.sun.com/j2se/1.5.0/docs/api/>. Acesso em: nov. 2006.

TABOADA, G. L.; TOURINO, J.; DOALLO, R. **Performance Modeling and Evaluation of Java Message-Passing Primitives on a Cluster.** Lecture Notes in Computer Science, Berlin, v. 2840, p. 29-36, 2003.