Avaliação de Desempenho na Recuperação de Imagens Distribuída

Eduardo Moschetta, Fernando S. Osório,
Gerson Geraldo H. Cavalheiro

Programa Interdisciplinar de Pós-Graduação em Computação Aplicada
Centro de Ciências Exatas e Tecnológicas
Universidade do Vale do Rio dos Sinos
{eduardom,osorio,gersonc}@exatas.unisinos.br

Introdução

Na área de processamento gráfico, sistemas de recuperação de imagens (CBIR) têm sido desenvolvidos para os mais diversos fins. Tais sistemas são programados para efetuar buscas de imagens ou fragmentos dessas em bases de dados, e retornar as regiões de imagens mais similares às procuradas. O critério de similaridade utilizado varia entre sistemas, podendo ser através de distribuição de cores, formas de objetos, texturas etc. Um exemplo desses sistemas pode ser visto em [DAS 97].

Entretanto, esses sistemas geram uma grande carga computacional, inviabilizando soluções que manipulam grande quantidade de imagens em arquiteturas monoprocessadas. Soluções que exploram aglomerados de computadores e a programação de alto desempenho apontam uma alternativa viável para estes problemas.

Inserido na classe de aplicações trivialmente paralelizáveis, o sistema de recuperação de imagens implementado nesse trabalho explora o paralelismo hídrido da arquitetura, dividindo a carga de trabalho entre nodos de um aglomerado e criando múltiplos fluxos de execução em cada nó participante.

O restante desse documento apresenta uma breve descrição do reconhecedor de imagens, critérios de similaridade utilizados, a solução concorrente empregada e uma avaliação de desempenho realizada em máquinas SMP e aglomerados de computadores.

Reconhecedor de Imagens

O núcleo do reconhecedor de imagens é responsável por percorrer todas as imagens de um banco de dados e coletar regiões dessas imagens similares a uma imagem exemplo (fragmento). É composto por uma cabeça de leitura, que varre a imagem, e um algoritmo de *matching*, o qual verifica a semelhança entre regiões e o fragmento segundo algum critério de similaridade previamente estabelecido.

A cada interação, a cabeça de leitura, com dimensões iguais ao fragmento procurado, é deslocada um pixel, quando então é aplicado o algoritmo de *matching*, que verifica a similaridade entre o fragmento e a região da imagem sobreposta pela cabeça de leitura. Esta é movimentada pixel a pixel de forma a percorrer todas regiões possíveis da imagem.

O algoritmo de *matching*, sob qualquer critério de similaridade, retorna um erro (diferença) de similaridade entre o fragmento e a região da imagem sobreposta no mo-

mento. A região é coletada pelo núcleo somente se esse erro encontra-se abaixo de um limite máximo de erro definido pelo usuário.

Critérios de similaridade

Existem muitas técnicas de *matching*: Antani [ANT 2002] apresenta um resumo dos principais sistemas e técnicas utilizadas atualmente na classificação e recuperação de imagens. Na ocasião de publicações anteriores desse trabalho [MOS 2002], apenas dois algoritmos de *matching* haviam sido implementados: comparação ponto a ponto e comparação de histogramas. Ambos algoritmos permitem a comparação nos sistemas de cores RGB ou HSV.

Atualmente, a ferramenta conta com mais dois algoritmos de *matching*: comparação ponto a ponto e de histogramas em múltiplas escalas. Esses algoritmos complementam os de única escala, realizando comparações com o fragmento em diferentes escalas.

Observa-se, entretanto, que a escolha desses algoritmos deu-se por sua simplicidade quanto à implementação, uma vez que a prioridade desse trabalho é encontrar estratégias de realizar uma busca eficiente no banco de imagens.

Solução concorrente

A implementação concorrente utiliza técnicas de paralelismo híbrido, considerando diferentes níveis de concorrência a serem explorados. O primeiro nível, de granularidade mais grossa, explora a concorrência do núcleo, permitindo que múltiplas imagens sejam processadas simultaneamente. O segundo, de granulosidade mais fina, explora a concorrência da cabeça de leitura, viabilizando o *matching* em várias regiões concorrentemente.

O paralelismo do primeiro nível é idealizado através da divisão do banco de imagens entre os nodos do aglomerado. Muitas vezes, entretanto, essa divisão é realizada de forma irregular, geralmente determinada por um banco com imagens de tamanhos variados. Nesse cenário, é implementado um balanceamento de carga, que detecta nodos ociosos e migra (rouba) imagens de nodos carregados para esses nodos.

A concorrência do segundo nível é realizada internamente em cada nó. A cabeça de leitura é decomposta em múltiplas cabeças de leitura, representadas por *threads*, que percorrem uma ou mais imagens de forma concorrente. O paralelismo nesse nível pode ser obtido caso o nó seja SMP. Além disso, a utilização de *threads* permite que parte do tempo gasto em comunicações seja sobreposto por computação efetiva [VAL 90].

Análise de Desempenho

Uma avaliação de desempenho foi realizada com vistas a validar a solução concorrente descrita. Em relação a [MOS 2002], os experimentos compreenderam novos algoritmos de *matching* e um aglomerado com maior número de computadores. Os experimentos foram realizados sobre um banco de dados consistindo de 505 imagens de diferentes dimensões, tendo em média 571x583 pixels e um desvio-padrão de 150x154

pixels [NEN 96], fisicamente duplicado entre os nodos do aglomerado; e sobre fragmentos de diferentes tamanhos – pequeno (135x135), médio (272x272) e grande (540x540) – a fim de testar a execução sobre diferentes granularidades.

Para fins de comparação, a Tabela 1 mostra os tempos de busca seqüencial dos três fragmentos com diferentes algoritmos de *matching*, obtidos em uma máquina monoprocessada (PIII 800MHz, 256Mb RAM). Observa-se que os tempos de processamento são elevados, com exceção ao fragmento grande, cujas buscas têm imagens de dimensões inferiores descartadas do processamento.

Tabela 1: Tempos de execução seqüencial

Algoritmo	Pequeno	Médio	Grande
Ponto a Ponto RGB	365.673 s	736.058 s	21.697 s
Ponto a Ponto HSV	441.309 s	778.648 s	22.040 s
Histogramas RGB	3.921 s	3.463 s	66 s
Histogramas HSV	10.823 s	9.362 s	164 s
Histogramas Escalas RGB	6.778 s	3.487 s	66 s
Histogramas Escalas HSV	19.907 s	9.448 s	165 s

Esses mesmos experimentos foram realizados em uma máquina SMP (1GHz, 512Mb RAM) e em um aglomerado de 6 nodos bi-processados (6 × PIII 1GHz, 512Mb RAM), documentados nas Tabelas 2 e 3, respectivamente. As tabelas contêm os tempos de execução e os respectivos ganhos obtidos com o paralelismo utilizado. Observa-se que em ambas arquiteturas os ganhos obtidos foram maiores que o de processadores disponíveis. Isso deve-se ao fato de que a máquina monoprocessada utilizada nos testes é de menor poder computacional que as utilizadas nas execuções concorrentes.

Tabela 2: Tempos de execução em uma máquina SMP

Fragmento	Pequeno		Médio		Grande	
Algoritmo	Tempo	Ganho	Tempo	Ganho	Tempo	Ganho
Ponto a Ponto RGB	142.248 s	2,57	286.573 s	2,56	9.004 s	2,41
Ponto a Ponto HSV	194.598 s	2,26	340.783 s	2,28	9.616 s	2,29
Histogramas RGB	1.584 s	2,47	1.649 s	2,10	58 s	1,14
Histogramas HSV	4.522 s	2,39	4.475 s	2,09	150 s	1,09
Hist. Escalas RGB	3.081 s	2,20	1.656 s	2,11	58 s	1,14
Hist. Escalas HSV	8.775 s	2,27	4.488 s	2,10	156 s	1,06

Os resultados nos permitiram validar a implementação realizada nos seguintes aspectos: a execução *multithreading* contribui com um bom ganho de desempenho em relação à seqüencial; a execução distribuída (com paralelismo híbrido) obteve igualmente bom ganho de desempenho, validando principalmente o algoritmo de balanceamento de carga, que executou sob cargas de trabalho iniciais irregularmente divididas.

Em relação a [MOS 2002], pode-se dizer que a aplicação é escalável, uma vez que os ganhos obtidos permaneceram bons em relação ao número de processadores utilizados. Além disso, os ganhos similares nos diferentes algoritmos de *matching* nos permitem concluir que o desempenho do aplicação independe da carga de trabalho global.

Tabela 3: Tempos de execução em um aglomerado de computadores

Fragmento	Pequeno		Médio		Grande	
Algoritmo	Tempo	Ganho	Tempo	Ganho	Tempo	Ganho
Ponto a Ponto RGB	28.478 s	12,84	57.290 s	12,85	1.996 s	10,87
Ponto a Ponto HSV	39.214 s	11,26	70.105 s	11,11	2.187 s	10,08
Histogramas RGB	268 s	14,63	274 s	12,63	11 s	6,00
Histogramas HSV	771 s	14,03	755 s	12,40	29 s	5,66
Hist. Escalas RGB	527 s	12,86	285 s	12,24	12 s	5,50
Hist. Escalas HSV	1.515 s	13,13	778 s	12,14	31 s	5,32

Conclusões

Este artigo apresenta a implementação concorrente de um algoritmo de recuperação de imagens. Diferentes aspectos dessa implementação foram destacados, tais como os diferentes níveis de granularidade e estratégias para a exploração eficiente do hardware. As análises de resultados realizadas são encorajantes, motivando a continuação do trabalho na área de reconhecimento de imagens.

Um aspecto importante para ser comentado é que embora o problema discutido aqui seja de importância real, a implementação foi realizada com o objetivo de validar conceitos e estratégias a serem utilizadas na implementação de um ambiente de programação concorrente: Anahy [CAV 2003].

Referências

[ANT 2002] ANTANI, S.; KASTURI, R.; RAMESH, J. A survey on the use of pattern recognition methods for abstraction, indexing and retrieval of images and video. **Pattern Recognition**, v.35, p.945–965, 2002.

[CAV 2003] CAVALHEIRO, G. G. H.; REAL, L. C. V.; DALL'AGNOL, E. C. Uma biblioteca de processos leves para a implementação de aplicações altamente paralelas. In: IV WORKSHOP EM SISTEMAS COMPUTACIONAIS DE ALTO DESEMPENHO, São Paulo, SP. **Anais...** [S.l.: s.n.], 2003. A ser publicado.

[DAS 97] DAS, M.; RISEMAN, E.; DRAPER, B. Focus: searching for multi-colored objects in a diverse image database. In: IEEE CONF. ON COMP. VIS. AND PATTERN RECOGNITION, Porto Rico. **Anais...** [S.l.: s.n.], 1997.

[MOS 2002] MOSCHETTA, E.; OSóRIO, F. S.; CAVALHEIRO, G. G. H. Reconhecimento de imagens em aplicações críticas. In: III WORKSHOP EM SISTEMAS COMPUTACIONAIS DE ALTO DESEMPENHO, Vitória, ES. **Anais...** [S.l.: s.n.], 2002.

[NEN 96] NENE, S.; NAYAR, S.; MURASE, H. **Columbia object image library**: coil. [S.l.]: Columbia University, 1996. (CUCS-006-96).

[VAL 90] VALIANT, L. G. A bridging model for parallel computation. **Communications of the ACM**, v.33, n.8, p.103–111, Aug. 1990.