Otimizando o tempo de execução no processamento de imagens

Henrique Miyamoto e Thiago Benites

I. Contextualização

Threads, assim como processos, são mecanismos que permitem que um programa execute ações de forma aparentemente simultânea. A diferença entre eles é que *threads* possuem área de memória compartilhada, o que não ocorre com processos [1]. Apresentamos uma comparação de desempenho entre diferentes métodos para aplicação de brilho em uma imagem. A mesma funcionalidade foi implementada de quatro maneiras: usando múltiplas *threadas*, usando multiprocessos, em uma única linha de execução, varrendo a matriz por linhas e por colunas. O objetivo é comparar e discutir os desempenhos de cada método a partir do tempo de execução de sistema de cada um deles.

II. Demonstração

Na comparação de desempenho, foram usadas imagens pequena (32x32 pixels), média (720x460) e grande (2592x1944). Foram medidos os tempos reais (tempo de relógio) e os tempos de usuário (tempo que a CPU gasta dentro dos processos).

Inicialmente, variou-se a quantidade de *threads* e processos em cada execução para otimizar esse número. Nesse procedimento, a imagem média foi usada como referência. A otimização do tempo de execução se dá para aproximadamente 800 *threads* e 1 processo, dentre os resultados avaliados.

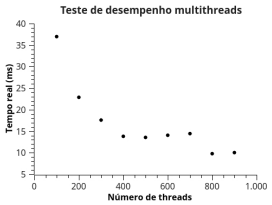

Fig. 1. Tempo real (ms) em função do número de *threads*.

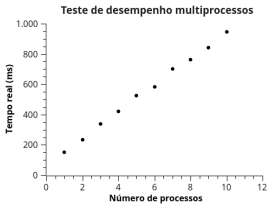

Fig. 2. Tempo real (ms) em função do número de multiprocessos.

* Os arquivos do projeto estão disponíveis em `https://github.com/miyamotohk/linguagem-processamento-imagem`.

A seguir, foram comparados os diferentes métodos de aplicação de brilho para cada imagem. Nesse teste, o número de *threads* e processos usados foi o número ótimo encontrado anteriormente.

TABLE I
Testes de desempenho para imagens pequena, média e grande

Método	Tamanho	Tempo real	Tempo de usuário
Multithreads	32x32	0,6650 ms	0,3650 ms
Multiprocessos	32x32	5,8270 ms	5,8280 ms
Varredura por colunas	32x32	0,0310 ms	0,0300 ms
Varredura por linhas	32x32	0,0210 ms	0,0200 ms
Multithreads	720x460	9,4800 ms	2,5760 ms
Multiprocessos	720x460	147,2950 ms	65,1070 ms
Varredura por colunas	720x460	9,2150 ms	9,2150 ms
Varredura por linhas	720x460	6,8940 ms	6,8930 ms
Multithreads	2592x1944	135,3250 ms	39,6660 ms
Multiprocessos	2592x1944	3819,1140 ms	637,9850 ms
Varredura por colunas	2592x1944	149,0920 ms	149,0950 ms
Varredura por linhas	2592x1944	142,3600 ms	142,3610 ms

III. Análise

Inicialmente, o número de *threads* e processos

Observa-se que o método com menor tempo de execução depende do tamanho da imagem utilizada. Para as imagens pequena e média, as varreduras em uma única linha de execução (por linha e por coluna) são as que apresentam melhor desempenho. No entanto, ao aplicar a funcionalidade à imagem grande, o tempo de execução do método *multithread* mostra-se o menor dentre os comparados. Isso se deve provavelmente à solução de compromisso do método de múltiplas *threads*: há um gasto computacional relacionado à criação delas, que, para imagens pequenas, acaba prejudicando o desempenho; para imagens grandes, no entanto, esse gasto é compensado pelo ganho de tempo da execução paralela das *threads*. No caso da implementação com múltiplos processos, seu desempenho é sempre inferior ao dos demais processos. Uma possível explicação para isso é o gasto necessário para criação da área de memória compartilhada.

Referências

[1] MITCHELL, Mark; OLDHAM, Jeffrey e SAMUEL, Alex. *Advanced Linux Programming*. Indianopolis: New Riders, 2001.