Projeto Final - Trabalho Prático 2

Disciplina: Sistemas Embarcados
Turma: 590
Professor: Sergio Johann Filho
Aluno: Társio Onofrio Cardoso da Silva

Introdução

Este projeto consiste na implementação paralela e distribuída de um algoritmo de processamento de imagens. Durante as etapas de processamento, o algoritmo a ser desenvolvido aplica dois filtros sobre cada ponto da imagem, sendo primeiramente aplicado um filtro gaussiano e posteriormente um filtro Sobel.

O sistema distribuído terá três tarefas: *source*, *worker* e *target*. A tarefa *source* é a fonte dos dados (imagem) que serão entregues para cada *worker*, esta recebe a solicitação de dados de cada *worker* e então envia para este. *Target* receberá os dados processados por cada um dos *workers*, será o alvo dessa tarefa. *Worker* é a tarefa central deste trabalho, após sua inicialização e ela faz solicitação de dados para a *source*, processa esses dados e envia para a *target*.

Desenvolvimento

Cada *worker* processará uma linha da imagem e não um bloco, se a imagem tem 128 por 128 pixels então ao invés de processar 32 blocos de 20 por 20 pixels (16 por 16 mais 4 pixels de borda na altura e largura), o sistema processará 124 (128 menos 2 pixels de borda à direita e a esquerda) blocos de 124 (128 menos 2 pixels de borda abaixo e acima) por 1, ou seja, processará a imagem linha a linha.

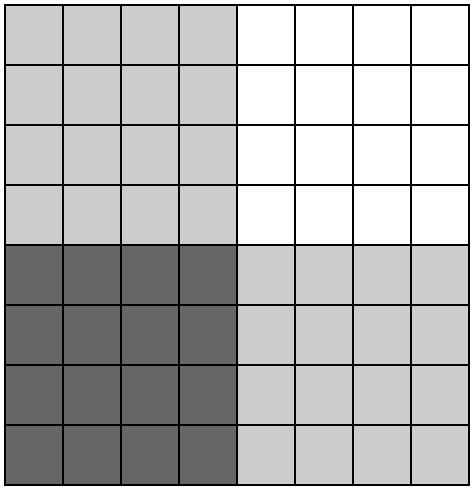


Imagem 8 por 8 dividida em 4 blocos de 4 por 4

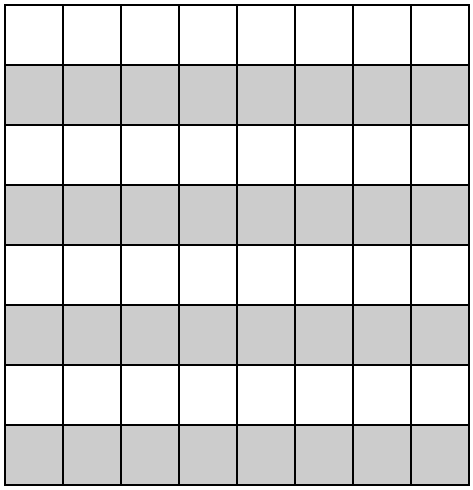


Imagem 8 por 8 dividida em 8 blocos de 8 por 1

O buffer deve ter tamanho compatível com a altura e largura do kernel dos filtros que tem tamanho diferente. O filtro de sobel tem kernel de tamanho 3 e o filtro de Gauss tem tamanho 5, devido a isso o tamanho de buffer escolhido no worker será de 128 por 5. Desta forma a linha central receberá a soma das convoluções.

x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x
x	x	x	x	x	x	x	x	x	x

*

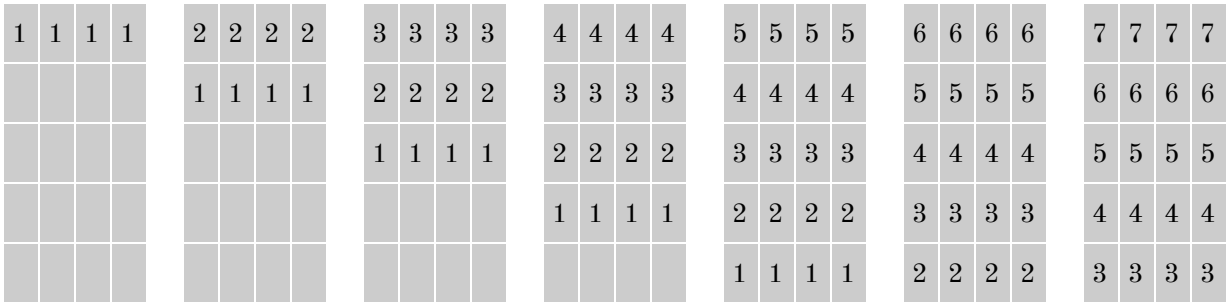
k	k	k	k	k
k	k	k	k	k
k	k	k	k	k
k	k	k	k	k
k	k	k	k	k

=

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	y	y	y	y	y	y	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Convolução entre uma imagem de 10 por 5 e um kernel de 5 por 5 com resultado a direita

Para evitar sobrecarga de comunicação devido ao envio 128 pixel por 5, 2 pixels de borda abaixo e acima da linha central, a cada solicitação a *worker* terá um FIFO de 5 posições (128 pixels por 5). *Source* enviará (128 pixels) a cada solicitação, cada solicitação será colocada no final do buffer e a cada nova solicitação o buffer será deslocado para a esquerda, a partir da 6 solicitação a primeira ou mais antiga é descartada.



Demonstração do deslocamento do buffer de tamanho de tamanho 5 com 4 pixels de largura

A tarefa *worker* enviará a linha central de cada processamento, 128 pixels ao centro do buffer, para a *target* que receberá os dados, colocará cada linha na devida posição e imprimirá os resultados quando não houver mais dados para processar.

Para facilitar o gerenciamento do envio e recepção dos blocos de memória nas tarefa *source* e *target*, haverá um ponteiro posicionado no início respectiva linha de cada worker. Tanto o envio quanto a recepção de dados no *source* e *target* ocorreram diretamente nesses ponteiros.

Resultados

Abaixo apresentamos os resultados da simulação:

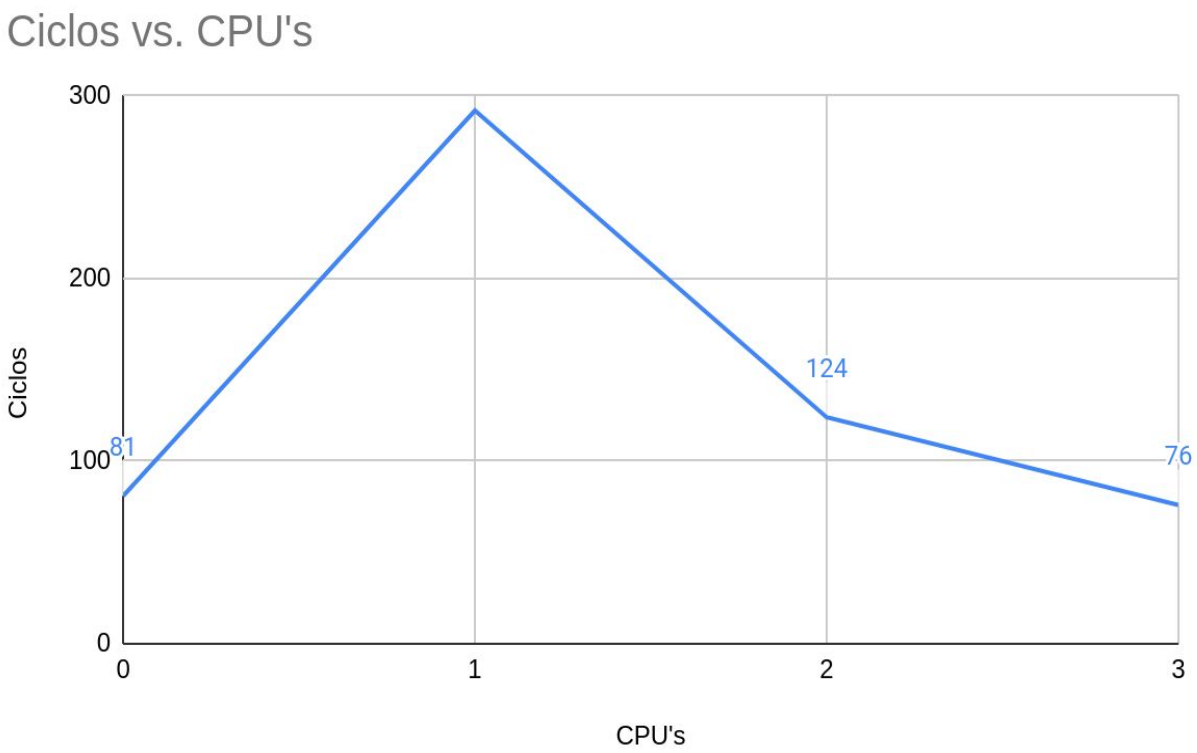


Gráfico Ciclos vs CPU/Workers na execução do sistema. CPU's 0 refere-se ao tempo de processamento local.

O tempo de execução do processamento distribuído cai agressivamente na medida em que são adicionados novos nós ao sistema com a tarefa *worker*. A cada nó o tempo cai aproximadamente pela metade.

Infelizmente não foi possível executar com 4 ou mais nós. O processamento para de funcionar após algumas comunicações.

Tutorial

Os arquivos do trabalho estão na pasta usr-tp2, dentro dessa pasta encontramos os diretórios relevantes:
noc-app: pasta com a aplicação desenvolvida nesse projeto.
output: pasta com a saída de cada um dos resultados deste trabalho