MC970/MO644 - Programação Paralela
Laboratório 11

Professor: Guido Araújo
Monitor: Rafael Cardoso Fernandes Sousa

# Filter Smoothing and 1D Convolution

Este laboratório é dividido em três etapas:

1. 1D Convolution using Shared Memory
2. Filter Smoothing using Shared Memory
3. Performance Analysis

## Enunciado

Para paralelizar os trabalhos, deve-se utilizar CUDA C. Os dois programas devem fazer o uso da shared memory.

### Convolution 1D

Para paralelizar o programa 1D Convolution, deve-se seguir o Design 1 do slide Convolution. Conforme:

- Design 1: The size of each thread block matches the size of an output tile
  - All threads participate in calculating output elements
  - Some threads need to load more than one input element into the shared memory

A computação que deve ser movida para a GPU é a seguinte:

```
void Convolution1D(int *N, int *P, int *M, int n) {
  int i, j;

  for(i=0; i < n; i++){
      int Pvalue = 0;
      int N_start_point = i - ((Mask_Width-1)/2);
      for(j = 0; j < Mask_Width; j++){
          if(N_start_point+j >=0 && N_start_point+j < n){
              Pvalue += N[N_start_point+j]*M[j];
          }
      }
      P[i] = Pvalue;
  }
}
```

As entradas desta estapa estão em formato texto. A primeira linha se refere a quantidade de elementos e, a segunda linha, descreve os elementos. Exemplo:

```
5
12 1 60 11 77
```

Observação: Coloquei uma máscara grande para aumentar a quantidade de operações realizadas por output.

## Filter Smoothing

A computação que deve ser movida para a GPU é a seguinte:

```
void Smoothing_CPU_Serial(PPMImage *image, PPMImage *image_copy) {
  int i, j, y, x;
  int total_red, total_blue, total_green;

  for (i = 0; i < image->y; i++) {
      for (j = 0; j < image->x; j++) {
          total_red = total_blue = total_green = 0;
          for (y = i - ((MASK_WIDTH-1)/2); y < (i + ((MASK_WIDTH-1)/2)); y++) {
              for (x = j - ((MASK_WIDTH-1)/2); x < (j + ((MASK_WIDTH-1)/2)); x++) {
                  if (x >= 0 && y >= 0 && y < image->y && x < image->x) {
                      total_red += image_copy->data[(y * image->x) + x].red;
                      total_blue += image_copy->data[(y * image->x) + x].blue;
                      total_green += image_copy->data[(y * image->x) + x].green;
                  } //if
              } //for z
          } //for y
          image->data[(i * image->x) + j].red = total_red / (MASK_WIDTH*MASK_WIDTH);
          image->data[(i * image->x) + j].blue = total_blue / (MASK_WIDTH*MASK_WIDTH);
          image->data[(i * image->x) + j].green = total_green / (MASK_WIDTH*MASK_WIDTH);
      }
  }
}
```

Mais detalhes: http://erad.dc.ufscar.br/problema.pdf. As entradas desta etapa estão no formato PPM, logo as cores são apenas RGB.

Os inputs consistem em 3 imagens com as seguintes resoluções: 720p, 1080p e 4k, todas no formato PPM.

## Performance Analysis

É necessário preencher, para cada um dos dois programas, 1D Convolution e Filtro Smoothing, uma Tabela no formato da Tabela .1.

Tabela .1: Speedup Filter Smoothing and 1D Convolution

| Entrada | CPU_Serial | GPU_NOSharedMemory | GPU_SharedMemory | Speedup (CPU/GPUSM) |
|---|---|---|---|---|
| Arq1.in | | | | |
| Arq2.in | | | | |
| Arq3.in | | | | |

Tabela .2: Only for Filter Smoothing

| | BLOCK_SIZE=8x8 | BLOCK_SIZE=14x14 | BLOCK_SIZE=15x15 | BLOCK_SIZE=16x16 | BLOCK_SIZE=32x32 |
|---|---|---|---|---|---|
| MASK_WIDTH=5 | | | | | |
| MASK_WIDTH=7 | | | | | |
| MASK_WIDTH=9 | | | | | |
| MASK_WIDTH=11 | | | | | |
| MASK_WIDTH=13 | | | | | |

Por fim, para o Filtro Smoothing é necessário preencher a Tabela .2. O slide Convolution, que está disponível no site de monitoria, mostra como é preenchido esta Tabela para 1DConvolution. MOSTRE COMO CHEGOU NO RESULTADO.

Observação: Os resultados devem ser incluídos em formato de comentário nos seus respectivos arquivos fontes.

# Testes e Resultado

Para compilar o seu programa, basta entrar no servidor mo644 ou parsusy, a partir do serviço ssh do IC, e digitar o comando **/usr/local/cuda-7.5/bin/nvcc programa.cu -o programa**. Para executar o 1D Convolution, basta digitar **./p < arq$.in > arq$.out**. Para executar o Filter Smoothing, basta digitar **./p arq$.ppm > out$.ppm**.

Não haverá comparação de Speedup na submissão dos Trabalhos. O Parsusy irá comparar apenas o output.

# Submissões

O número máximo de submissões é de 10. Antes de submeter seu programa, faça testes usando o comando diff do Linux, exemplo: diff gpu_out.ppm cpu_out.ppm.

## Compilação e Execução

O ParSuSy irá compilar o seu programa através do compilador nvcc.

# Links Úteis

https://www.vivaolinux.com.br/dica/Utilizando-o-comando-scp.
https://sites.google.com/site/mo644mc970/slides.