

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

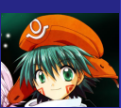
Matlab com
cuda

Bibliografia

Utilização da unidade processamento gráfico para propósito geral (GPGPU)

Rafael Guedes Lang
Anderson Gonçalves Marco

26 de fevereiro de 2009



Sumário

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

1 Introdução

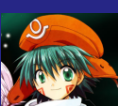
2 Comparação entre GPUs e CPUs

3 Arquitetura de GPUs

4 Cuda

5 Matlab com cuda

6 Bibliografia



O que é (GPGPU) ?

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

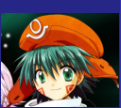
Cuda

Matlab com
cuda

Bibliografia

E a utilização da placa de vídeo como um co-processador matemático.

Através criação de rotinas que rodam dentro do processador da placa de vídeo(gpu),por meio de bibliotecas como: Brook, Close to Metal(ATI), Sh, OpenCL(Apple), CUDA(NVIDIA)



Historia das placas de vídeo 3D

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

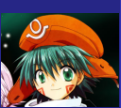
Matlab com
cuda

Bibliografia

Foram criadas inicialmente para aumentar a qualidade gráfica dos jogos (Hoje qualquer jogo 3D precisa de uma placa 3D).

Antes de 1999 as placa aceleradoras 3D, domesticas, cuidavam basicamente da aplicação de texturas em superfícies de polígonos. Com a Geforce 256 as placas de vídeo passaram a incorporar outras funções, como transformações de vértices, que antes eram desempenhadas pela FPU do processador central do computador.

Para isto o processador da placa de vídeo precisou evoluir para uma gpu(Unidade de processamento gráfico), tornando-se extremamente eficaz em operações com ponto flutuante.



Sumário

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

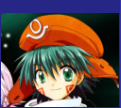
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

- 1 Introdução
- 2 Comparação entre GPUs e CPUs
- 3 Arquitetura de GPUs
- 4 Cuda
- 5 Matlab com cuda
- 6 Bibliografia



Comparação entre Processadores Gráficos e Processadores Centrais

Utilização da unidade processamento gráfico para propósito geral (GPGPU)

Rafael Guedes Lang
Anderson Gonçalves
Marco

Introdução



Comparação entre GPUs e CPUs

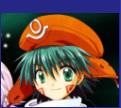
Arquitetura de GPUs

Cuda

Matlab com cuda

Bibliografia

GPUs	CPUs
	
Propósito específico (lidar com floats)	Propósito genérico
Unidades de processamento simples e em grande quantidade	Unidades de processamento complexas e em pequena quantidade
Ruim em previsão de desvio (if,else)	Bom em previsão de desvio
Não segue completamente o padrão para pontos flutuantes, IEEE 754	Segue o padrão IEEE 754
Paralelismo agressivo	Paralelismo modesto



Comparação entre Processadores Gráficos e Processadores Centrais

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

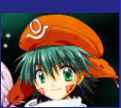
Matlab com
cuda

Bibliografia

Estas diferenças conferem as GPUs um ganho brutal em operações aritméticas de floats.

O Gráfico do slide seguinte confirma isto.

Este gráfico está um pouco desatualizada com relação a o desempenho das GPUs, a nova gpu GTX280 da Nvidia alcança a marca de 1 Teraflop



Comparação do crescimento dos Gflops, ao longo do tempo, entra as GPUs Nvidia, ATI e CPUs Intel

Utilização da unidade processamento gráfico para propósito geral (GPGPU)

Rafael Guedes Lang Anderson Gonçalves Marco

Introdução

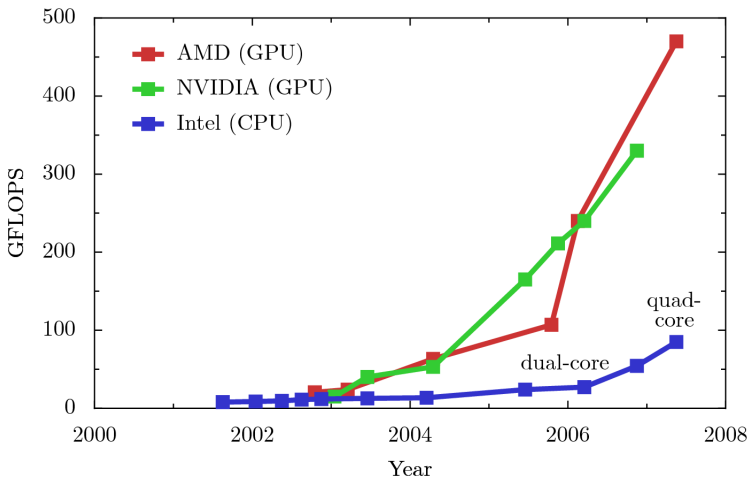
Comparação entre GPUs e CPUs

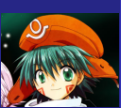
Arquitetura de GPUs

Cuda

Matlab com cuda

Bibliografia





Sumário

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

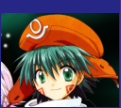
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

- 1 Introdução
- 2 Comparação entre GPUs e CPUs
- 3 Arquitetura de GPUs**
- 4 Cuda
- 5 Matlab com cuda
- 6 Bibliografia



Arquitetura de GPUs

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

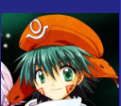
Cuda

Matlab com
cuda

Bibliografia

A programação em cuda exige que se conheça um pouco da arquitetura da GPU, pode-se simplificar a arquitetura de uma GPU para seguinte forma:

Uma GPU e dividida em blocos cada bloco possui n processadores, cada processador possui uma memoria local, muito rápida, cada bloco possui uma memoria regional(**shared memory**), um pouco mais lenta, que é compartilhada entre os processadores pertencentes ao bloco, a gpu possui uma memoria global(**device memory**), que é mais lenta que a memoria regional de um bloco, compartilha entre todos os processadores da gpu existem também duas memorias rápidas acessíveis a todos o processadores da gpu, mas que apenas a CPU pode escrever, a memoria de texturas(**texture memory**) e a memoria constante(**constant memory**).



Esquema da arquitetura de GPUs

Utilização da unidade processamento gráfico para propósito geral (GPGPU)

Rafael Guedes Lang Anderson Gonçalves Marco

Introdução

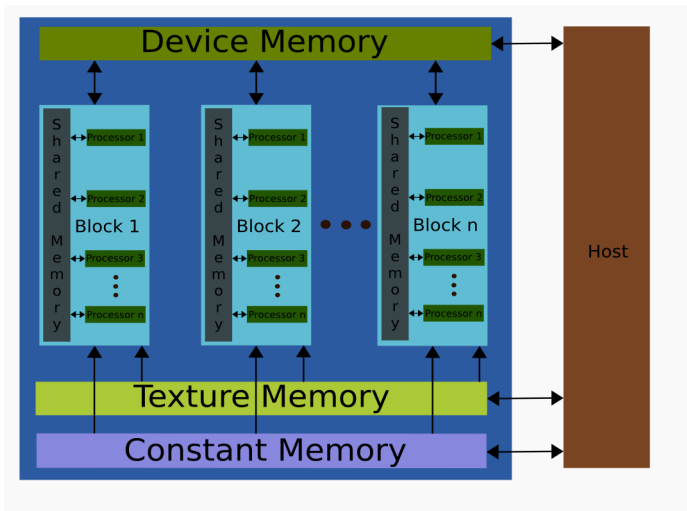
Comparação entre GPUs e CPUs

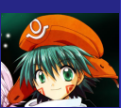
Arquitetura de GPUs

Cuda

Matlab com cuda

Bibliografia





Sumário

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

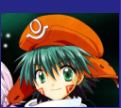
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

- 1 Introdução
- 2 Comparação entre GPUs e CPUs
- 3 Arquitetura de GPUs
- 4 Cuda**
- 5 Matlab com cuda
- 6 Bibliografia



Oque e Cuda?

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

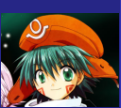
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

- E uma biblioteca e um compilador para criação de rotinas para GPUs Nvidia.



Oque e Cuda?

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

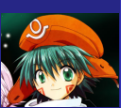
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

- E uma biblioteca e um compilador para criação de rotinas para GPUs Nvidia.
- Esta API esta num nível intermediário entre as APIs de baixo nível (Cg Toolkit) e alto nível (Brooks), sendo necessário ainda algum conhecimento de como a GPU funciona internamente para poder utiliza-la adequadamente.



O que é Cuda?

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

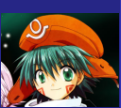
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

- É uma biblioteca e um compilador para criação de rotinas para GPUs Nvidia.
- Esta API está num nível intermediário entre as APIs de baixo nível (Cg Toolkit) e alto nível (Brooks), sendo necessário ainda algum conhecimento de como a GPU funciona internamente para poder utilizá-la adequadamente.
- Entretanto ela é uma API estável, e vem sendo utilizada em muitos projetos científicos.



Oque e Cuda?

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

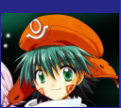
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

- E uma biblioteca e um compilador para criação de rotinas para GPUs Nvidia.
- Esta API esta num nível intermediário entre as APIs de baixo nível (Cg Toolkit) e alto nível (Brooks), sendo necessário ainda algum conhecimento de como a GPU funciona internamente para poder utiliza-la adequadamente.
- Entretanto ela e uma API estável, e vem sendo utilizada em muitos projetos científicos.
- Possui uma grande comunidade e boa documentação.



Requisitos para instalar Cuda

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

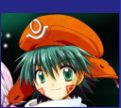
Cuda

Matlab com
cuda

Bibliografia

Uma placa de vídeo com GPU G80 (Gerfoce 8, Quadro) ou posterior.

Ter instalado no computador o driver da placa de vídeo, recomenda-se as ultimas versões do driver.



Compilando programas em Cuda

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

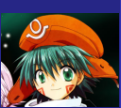
Matlab com
cuda

Bibliografia

Para se compilar um programa em cuda deve-se usar o comando `nvcc arquivo-fonte.cu -o arquivo-de-saida`.

Não e necessário uma placa Nvidia para se compilar, mais e necessário para se rodar o executável.

Os arquivos fontes cuda tem a extensão `*.cu`.



Primeiro programa em Cuda

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

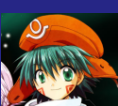
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

No próximo slide sera visto um simples programa, oque ele faz e pegar um vetor elevar todas as componentes deste vetor ao quadrado e multiplicar por um escalar.



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

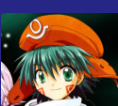
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

```
1 // incrementArray.cu
2 #include <assert.h>
3 #include <cuda.h>
4 #include <stdio.h>
5 __global__ void mult_vet(float *vet,float *vetr,float mul)
6 {
7     int idx = blockIdx.x * blockDim.x + threadIdx.x;
8     vetr[idx]=vet[idx]*vet[idx]*mul;
9 }
10
11
12 int main(void)
13 {
14     float *a_h, *b_h;           // pointers to host memory
15     float *a_d,*b_d;           // pointer to device memory
16     int i, N = 256;
17     size_t size = N*sizeof(float);
18     // allocate arrays on host
19     a_h = (float *)malloc(size);
20     b_h = (float *)malloc(size);
21     // allocate array on device
22     cudaMalloc((void **) &a_d, size);
23     cudaMalloc((void **) &b_d, size);
24     // initialization of host data
25     for (i=0; i<N; i++) a_h[i] = (float)45;
26     for (i=0; i<N; i++) b_h[i] = (float)0;
27     a_h[251]=56.0f;
28     // copy data from host to device
29     cudaMemcpy(a_d, a_h, sizeof(float)*N, cudaMemcpyHostToDevice);
30     cudaMemcpy(b_d, b_h, sizeof(float)*N, cudaMemcpyHostToDevice);
31     int blockSize = 16;
```



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

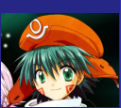
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

```
32     int nBlocks = 16;
33     mult_vet <<< nBlocks, blockSize >>> (a_d,b_d,32.0f);
34     cudaMemcpy(b_h, b_d, sizeof(float)*N, cudaMemcpyDeviceToHost);
35     for (i=0; i < N; i++) printf("%f \n",b_h[i] );
36     free(a_h); free(b_h); cudaFree(a_d); cudaFree(b_d);
37
38 }
```



Explicando o código acima

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

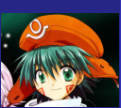
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

- Os vetores (ponteiros) `a_h` e `b_h` são alocados na memória do host (**memoria ram do computador**) e inicializados com um valor qualquer (**linhas 20, 21, 26 e 27**).
- Os vetores (ponteiros) `a_d` e `b_d` são alocados na memory device da gpu (**linhas 23, 24**), o conteúdo de `a_h` e `b_h` e copiado para `a_d` e `b_d` (**linhas 30, 31**).
- define-se e a quantidade de processos a serem executados na gpu (**linhas 32, 33**). O ideal seria que `block_Size` fosse um multiplo de 64 e `nBlocks` fosse ≥ 64 (a Nvidia recomenda ≥ 100), pois neste arranjo é que se obtem o maximo de performance.



Explicando o código acima

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

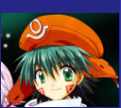
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

- Chama-se a função a ser executada na gpu.
- Copia-se valor do vetor `b_d` para `b_h` (linha 35).
- Desaloca-se os vetores (linha 37).



Aumentando o desempenho

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

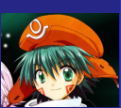
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

- Repare que no vetor `a_d` os processadores da gpu fazem apenas operações de leitura não de escrita.



Aumentando o desempenho

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

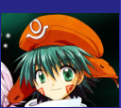
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

- Repare que no vetor `a_d` os processadores da gpu fazem apenas operações de leitura não de escrita.
- Colocar este vetor então na memory texture ou na memory constant, aumenta a velocidade de transferência, aumentando o desempenho.



Aumentando o desempenho

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

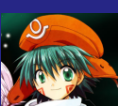
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

- Repare que no vetor `a_d` os processadores da gpu fazem apenas operações de leitura não de escrita.
- Colocar este vetor então na memory texture ou na memory constant, aumenta a velocidade de transferência, aumentando o desempenho.
- No próximo slide uma modificação usando memory texture.



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

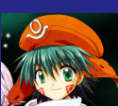
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

```
1  #include <assert.h>
2  #include <cuda.h>
3  #include <stdio.h>
4  texture<float, 1, cudaReadModeElementType> texRef;//novo
5  __global__ void mult_vet(float *vetr,float mul)
6  {
7      int idx = blockIdx.x * blockDim.x + threadIdx.x;//ID da Thread
8      vetr[idx]= tex1Dfetch(texRef, idx)*tex1Dfetch(texRef, idx)*mul;//novo
9  }
10
11
12 int main(void)
13 {
14     float *a_h, *b_h;           // pointers to host memory
15     float *a_d,*b_d;           // pointer to device memory
16     int i, N = 256;
17     size_t size = N*sizeof(float);
18     // allocate arrays on host
19     a_h = (float *)malloc(size);
20     b_h = (float *)malloc(size);
21     // allocate array on device
22     cudaMalloc((void **) &a_d, size);
23     cudaMalloc((void **) &b_d, size);
24     // initialization of host data
25     for (i=0; i<N; i++) a_h[i] = (float)45;
26     for (i=0; i<N; i++) b_h[i] = (float)0;
27     a_h[251]=56.0f;
28     // copy data from host to device
29     cudaMemcpy(a_d, a_h, sizeof(float)*N, cudaMemcpyHostToDevice);
30     cudaMemcpy(b_d, b_h, sizeof(float)*N, cudaMemcpyHostToDevice);
31     cudaBindTexture(0,texRef, a_d, sizeof(float)*N);//novo
```



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

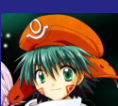
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

```
32     int blockSize = 16;
33     int nBlocks = 16;
34     mult_vet <<< nBlocks, blockSize >>> (b_d,32.0f);
35     cudaMemcpy(b_h, b_d, sizeof(float)*N, cudaMemcpyDeviceToHost);
36     for (i=0; i < N; i++) printf("%f \n",b_h[i] );
37     cudaUnbindTexture(texRef);//novo
38     free(a_h); free(b_h); cudaFree(a_d); cudaFree(b_d);
39 }
40
```



Caso 1D

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

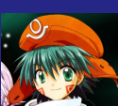
Cuda

Matlab com
cuda

Bibliografia

Explicando o código visto anteriormente

- Na linha 4 é criada um ponteiro para uma região da textura.
- Na linha 31 copia-se o conteúdo do vetor `a_d` para uma região da memory texture.
- Na linha 7 se acessa uma região da textura multiplica-se o valor desta região por ele mesmo e pela variável `mul`.
- Na linha 37 desaloca-se a região de textura para onde o vetor `a_d` foi copiado.



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

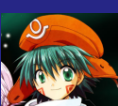
Matlab com
cuda

Bibliografia

Outras maneiras de usar a memoria de textura

- Alocar o vetor na estrutura de array do cuda, para então passar esta estrutura para a memoria de texturas, permite que a gpu realize algumas operações de maneira "automatica" sobre os dados deste array (como mapeamento do discreto para continuo nos indices, dos elementos, e interpolação dos pontos com os seus elementos vizinhos) que para alguns problemas, com que se trabalha na computação científica, podem ser uteis.
- Alocar a memoria em 2 dimensões, isto melhora o desempenho quando se trabalha com matrizes.

No próximo slide um exemplo, de como usar a estrutura de array do cuda.



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

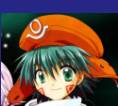
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

```
1  #include <assert.h>
2  #include <cuda.h>
3  #include <stdio.h>
4  texture<float, 1, cudaReadModeElementType> texRef;//novo
5  __global__ void mult_vet(float *vetr,float mul)
6  {
7      int idx = blockIdx.x * blockDim.x + threadIdx.x;//ID da Thread
8      vetr[idx]= tex1D(texRef,idx)*tex1D(texRef,idx)*mul;//novo
9  }
10 int main(void)
11 {
12     float *a_h, *b_h;           // pointers to host memory
13     float *b_d;                // pointer to device memory
14     int i, N = 256;
15     size_t size = N*sizeof(float);
16     // allocate arrays on host
17     a_h = (float *)malloc(size);
18     b_h = (float *)malloc(size);
19     // allocate array on device
20     cudaMalloc((void **) &b_d, size);
21     // initialization of host data
22     for (i=0; i<N; i++) a_h[i] = (float)i;
23     for (i=0; i<N; i++) b_h[i] = (float)0;
24     a_h[251]=56.0f;
25     // copy data from host to device
26     cudaArray *texArray = 0;
27     cudaChannelFormatDesc cf = cudaCreateChannelDesc<float>();
28     cudaMallocArray(&texArray, &cf,256,1);
29     cudaMemcpyToArray(texArray, 0,0, a_h, size, cudaMemcpyHostToDevice);
30     texRef.normalized = 0;
31     texRef.filterMode = cudaFilterModePoint;
```



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

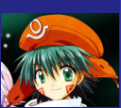
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

```
32     texRef.addressMode[0] = cudaAddressModeClamp;
33     cudaBindTextureToArray(texRef, texArray);
34     int blockSize = 16;
35     int nBlocks = 16;
36     mult_vet <<< nBlocks, blockSize >>> (b_d,32.0f);
37     cudaMemcpy(b_h, b_d, sizeof(float)*N, cudaMemcpyDeviceToHost);
38     for (i=0; i < N; i++) printf("%f \n",b_h[i] );
39     cudaUnbindTexture(texRef);//novo
40     free(a_h); free(b_h);  cudaFree(b_d);cudaFreeArray(texArray);
41 }
42
```



Caso 1D

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

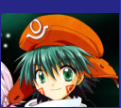
Cuda

Matlab com
cuda

Bibliografia

Explicando o código visto anteriormente

- Na linha 4 é criada um ponteiro para uma região da textura.
- Na linha 29 aloca-se espaço para um array cuda.
- Na linha 30 copia-se o conteúdo do vetor `a_h` para um array cuda.
- Na linha 31 diz como a região de textura deve ser mapeada, 0 para ser mapeada da forma normal, onde índices vão de 0 ao tamanho do vetor-1, diferente de 0 para ser mapeada na forma contínua, onde os índices vão de 0.0 a 1.0.
- Na linha 32, diz se a região de textura vai (`cudaFilterModeLinear`) ou não (`cudaFilterModePoint`) interpolar os valores passados a ela.



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

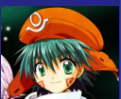
Cuda

Matlab com
cuda

Bibliografia

Cont ...

- Na linha 33 diz como a região de textura vai lidar com índices que extrapolam as dimensões do vetor (ou da matriz), colocadas em sua região.
- Na linha 34 copia-se o conteúdo do array cuda para a região de textura;
- Na linha 8 se acessa uma região da textura multiplica-se o valor desta região por ele mesmo e pela variável `mul`.
- Na linha 40 desaloca-se a região de textura para onde o vetor `a_d` foi copiado.
- Na linha 41 desaloca-se as variáveis alocadas dinamicamente.



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

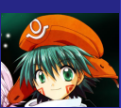
Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia



Caso 2D

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

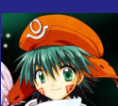
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

No próximo slide sera visto um exemplo de como alocar matrizes na memoria de textura.



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

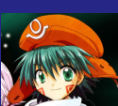
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

```
1  #include <assert.h>
2  #include <cuda.h>
3  #include <stdio.h>
4  texture<float, 2, cudaReadModeElementType> texRef;//novo
5  __global__ void mult_vet(float *vetr,float mul)
6  {
7      int idx = blockIdx.x * blockDim.x + threadIdx.x;//ID da Thread
8      // vetr[idx]= tex2D(texRef,threadIdx.x,blockIdx.x)*tex2D(texRef, threadIdx.x,blockIdx.x)*mul
9      vetr[idx]=tex2D(texRef, threadIdx.x,blockIdx.x);//novo
10 }
11
12
13 int main(void)
14 {
15     float *a_h, *b_h;           // pointers to host memory
16     float *b_d;                // pointer to device memory
17     int i, N = 256;
18     size_t size = N*sizeof(float);
19     // allocate arrays on host
20     a_h = (float *)malloc(size);
21     b_h = (float *)malloc(size);
22     // allocate array on device
23     cudaMalloc((void **) &b_d, size);
24     // initialization of host data
25     for (i=0; i<N; i++) a_h[i] = (float)i;
26     for (i=0; i<N; i++) b_h[i] = (float)0;
27     a_h[251]=56.0f;
28     // copy data from host to device
29     cudaArray *texArray = 0;
30     cudaChannelFormatDesc cf = cudaCreateChannelDesc<float>();
31     cudaMallocArray(&texArray, &cf, 16, 16);
```



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

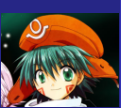
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

```
32     cudaMemcpyToArray(texArray, 0,0, a_h, size, cudaMemcpyHostToDevice);
33     texRef.normalized = 0;
34     texRef.filterMode = cudaFilterModePoint;
35     texRef.addressMode[0] = cudaAddressModeClamp;
36     texRef.addressMode[1] = cudaAddressModeClamp;
37     // texRef.addressMode[1] =cudaAddressModeWrap ;
38
39     cudaBindTextureToArray(texRef, texArray);
40     int blockSize = 16;
41     int nBlocks = 16;
42     mult_vet <<< nBlocks, blockSize >>> (b_d,32.0f);
43     cudaMemcpy(b_h, b_d, sizeof(float)*N, cudaMemcpyDeviceToHost);
44     for (i=0; i < N; i++) printf("%f \n",b_h[i] );
45     cudaUnbindTexture(texRef);//novo
46     free(a_h); free(b_h);  cudaFree(b_d);cudaFreeArray(texArray);
47 }
48
```



Constant Memory

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

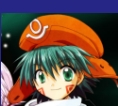
Matlab com
cuda

Bibliografia

É uma memória rápida vista por todos os processadores da gpu mas apenas o host pode escrever nela.

Limitações da Constant Memory

Ela é uma memória de alocação estática e atribuição estática.



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

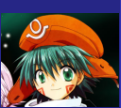
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

```
1  #include <assert.h>
2  #include <cuda.h>
3  #include <stdio.h>
4  __constant__ float a_c[255]={45.0f,70.0f};
5  __global__ void mult_vet(float *vetr,float mul)
6  {
7      int idx = blockIdx.x * blockDim.x + threadIdx.x;//ID da Thread
8      vetr[idx]= a_c[idx]*a_c[idx]*mul;//novo
9  }
10
11
12 int main(void)
13 {
14     float *b_h;           // pointers to host memory
15     float *b_d;           // pointer to device memory
16     int i, N = 256;
17     size_t size = N*sizeof(float);
18     // allocate arrays on host
19     b_h = (float *)malloc(size);
20     // allocate array on device
21     cudaMalloc((void **) &b_d, size);
22     // initialization of host data
23     for (i=0; i<N; i++) b_h[i] = (float)0;
24     // copy data from host to device
25     int blockSize = 16;
26     int nBlocks = 16;
27     mult_vet <<< nBlocks, blockSize >>> (b_d,32.0f);
28     cudaMemcpy(b_h, b_d, sizeof(float)*N, cudaMemcpyDeviceToHost);
29     for (i=0; i < N; i++) printf("%f \n",b_h[i] );
30     free(b_h); cudaFree(b_d);
31 }
```



32

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

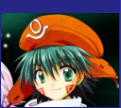
Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia



Comunicação entre processos

Utilização da unidade processamento gráfico para propósito geral (GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação entre GPUs e CPUs

Arquitetura de GPUs

Cuda

Matlab com cuda

Bibliografia

Como cuda segue o modelo de programação Memória Compartilha a comunicação entre processos se dá através da memória.

Para este fim usa-se a device memory e a shared memory.

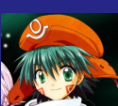
Device memory

É mais lenta usa-se para se fazer comunicação entre processos de diferentes blocos e funções da gpu.

Shared Memory

É mais rápida, usa-se para se fazer comunicação entre diferentes processos de um mesmo bloco.

No próximo slide um exemplo de comunicação utilizando a device memory.



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

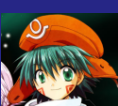
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

```
1  #include <assert.h>
2  #include <cuda.h>
3  #include <stdio.h>
4  texture<float, 1, cudaReadModeElementType> texRef;
5  __device__ float dev[256]; // novo
6  __global__ void mult_vet(float *vetr, float mul)
7  {
8      int idx = blockIdx.x * blockDim.x + threadIdx.x; // ID da Thread
9      vetr[idx] = tex1Dfetch(texRef, idx) * tex1Dfetch(texRef, idx) * mul; // novo
10     dev[idx] = vetr[idx] * 2.0f;
11 }
12
13 __global__ void mult_vet2(float *vetr) {
14     int idx = blockIdx.x * blockDim.x + threadIdx.x; // ID da Thread
15     vetr[idx] = dev[idx] * vetr[idx];
16 }
17
18 __global__ void mult_vet(float *vetr, float mul);
19 int main(void)
20 {
21     float *a_h, *b_h; // pointers to host memory
22     float *a_d, *b_d, *b_d2; // pointer to device memory
23     int i, N = 256;
24     size_t size = N * sizeof(float);
25     // allocate arrays on host
26     a_h = (float *) malloc(size);
27     b_h = (float *) malloc(size);
28     // allocate array on device
29     cudaMalloc((void **) &a_d, size);
30     cudaMalloc((void **) &b_d, size);
31     cudaMalloc((void **) &b_d2, size);
```



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

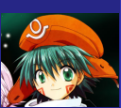
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

```
32 // initialization of host data
33 for (i=0; i<N; i++) a_h[i] = 7.3242f;
34 for (i=0; i<N; i++) b_h[i] = (float)0;
35 a_h[251]=6.0f;
36 // copy data from host to device
37 cudaMemcpy(a_d, a_h, sizeof(float)*N, cudaMemcpyHostToDevice);
38 cudaMemcpy(b_d, b_h, sizeof(float)*N, cudaMemcpyHostToDevice);
39 cudaBindTexture(0,texRef, a_d, sizeof(float)*N);//novo
40 int blockSize = 16;
41 int nBlocks = 16;
42 mult_vet <<< nBlocks, blockSize >>> (b_d,2.234f);
43 cudaThreadSynchronize();
44 mult_vet2 <<< nBlocks, blockSize >>>(b_d);
45 cudaMemcpy(b_h, b_d, sizeof(float)*N, cudaMemcpyDeviceToHost);
46 for (i=0; i < N; i++) printf("%f \n",b_h[i] );
47 cudaUnbindTexture(texRef);//novo
48 free(a_h); free(b_h); cudaFree(a_d); cudaFree(b_d);
49 }
50
51
```



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

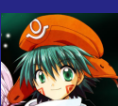
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

No próximo slide um exemplo de comunicação utilizando a shared memory.



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

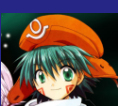
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

```
1  #include <assert.h>
2  #include <cuda.h>
3  #include <stdio.h>
4  texture<float, 1, cudaReadModeElementType> texRef;//novo
5  __global__ void mult_vet(float *vetr,float mul)
6  {
7      int idx = blockIdx.x * blockDim.x + threadIdx.x;//ID da Thread
8      __shared__ float shared[16];
9      if(threadIdx.x==0) shared[blockIdx.x]=(float)blockIdx.x;
10     __syncthreads(); //sincroniza as threads dos blocos
11     vetr[idx]= tex1Dfetch(texRef, idx)*tex1Dfetch(texRef, idx)*mul*shared[blockIdx.x];//novo
12 }
13
14
15 __global__ void mult_vet(float *vetr,float mul);
16 int main(void)
17 {
18     float *a_h, *b_h;           // pointers to host memory
19     float *a_d,*b_d;           // pointer to device memory
20     int i, N = 256;
21     size_t size = N*sizeof(float);
22     // allocate arrays on host
23     a_h = (float *)malloc(size);
24     b_h = (float *)malloc(size);
25     // allocate array on device
26     cudaMalloc((void **) &a_d, size);
27     cudaMalloc((void **) &b_d, size);
28     // initialization of host data
29     for (i=0; i<N; i++) a_h[i] = (float)45;
30     for (i=0; i<N; i++) b_h[i] = (float)0;
31     a_h[251]=56.0f;
```



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

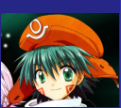
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

```
32 // copy data from host to device
33 cudaMemcpy(a_d, a_h, sizeof(float)*N, cudaMemcpyHostToDevice);
34 cudaBindTexture(0, texRef, a_d, sizeof(float)*N); //novo
35 int blockSize = 16;
36 int nBlocks = 16;
37 mult_vet <<< nBlocks, blockSize >>> (b_d, 32.0f);
38 cudaMemcpy(b_h, b_d, sizeof(float)*N, cudaMemcpyDeviceToHost);
39 for (i=0; i < N; i++) printf("%f \n", b_h[i]);
40 cudaUnbindTexture(texRef); //novo
41 free(a_h); free(b_h); cudaFree(a_d); cudaFree(b_d);
42 }
43
44
```



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

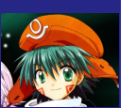
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

No próximo slide um exemplo de comunicação utilizando a shared memory com alocação dinâmica.



Tipos de rotinas

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

Matlab com
cuda

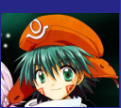
Bibliografia

Rotinas bloqueantes

São aquelas que esperam que todas as chamadas cuda antes dela tenham sido processadas para então poderem ser chamadas e quando chamadas ocupam todo processamento, só liberando o processador quando terminada sua execução.

Exemplos:

- `cudaMemcpy`.
- `cudaThreadSynchronize()` utilizada para sincronização.



Tipos de rotinas

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

Matlab com
cuda

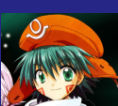
Bibliografia

Rotinas não bloqueantes

Permitem a execução de outras chamadas Cuda, não bloqueantes, paralelamente.

Exemplos:

- `cudaMemcpyAsync`.
- As rotinas definidas, por nos, que rodam na gpu.



Estruturas de dados

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

Matlab com
cuda

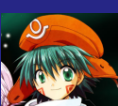
Bibliografia

Cuda possui estruturas de dados já prontas e permite a criação de novas estruturas de dados.

Estruturas de dados já definidas pelo cuda

`char1`, `uchar1`, `char2`, `uchar2`, `char3`, `uchar3`, `char4`, `uchar4`,
`short1`, `ushort1`, `short2`, `ushort2`, `short3`, `ushort3`, `short4`,
`ushort4`, `int1`, `uint1`, `int2`, `uint2`, `int3`, `uint3`, `int4`, `uint4`, `long1`,
`ulong1`, `long2`, `ulong2`, `long3`, `ulong3`, `long4`, `ulong4`, `float1`,
`float2`, `float3`, `float4`

São vetores onde a parte em verde e o tipo de dado e a parte em vermelho e o numero de componentes $[x(1),y(2),z(3),w(4)]$.



Estruturas de dados

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

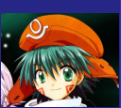
Cuda

Matlab com
cuda

Bibliografia

Exemplo

```
float4 posi;  
posi.x=3.0f;  
posi.y=10.0f;  
posi.z=2.0f;  
posi.w=-2.34f;
```



Criando estruturas de dados

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

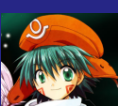
Cuda

Matlab com
cuda

Bibliografia

Exemplo de criação de uma estrutura de dados

```
typedef struct __align__(16){  
    float a;  
    float b;  
    float c;  
    float d;  
    float e;  
}vet;
```



Funções matemáticas otimizadas para GPU

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

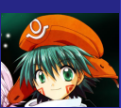
Matlab com
cuda

Bibliografia

As funções matemáticas, otimizadas para floats, são as mesmas encontradas na biblioteca `math.h` do C-ANSI, porem elas devem precedidas por `__`.

Exemplo:

CPU	GPU
<code>sinf(float x)</code>	<code>__sinf(float x)</code>
<code>cosf(float x)</code>	<code>__cosf(float x)</code>
<code>logf(float x)</code>	<code>__logf(float x)</code>



Funções matemáticas otimizadas para GPU

Utilização da
unidade
processamento
gráfico para
propósito
geral
(**GPGPU**)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

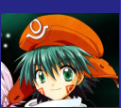
O uso da diretiva `-use_fast_math` na hora da compilação, transforma as funções não otimizadas(sem `__`) em otimizadas.

Exemplo:

```
nvcc arquivo-fonte.cu -use_fast_math -o arquivo-de-saida
```

Problemas das funções otimizadas

Elas possuem uma precisão menor que as não otimizadas.



Sumário

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

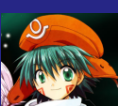
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

- 1 Introdução
- 2 Comparação entre GPUs e CPUs
- 3 Arquitetura de GPUs
- 4 Cuda
- 5 Matlab com cuda**
- 6 Bibliografia



Matlab com cuda

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

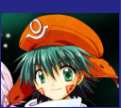
Cuda

Matlab com
cuda

Bibliografia

Oque e o Matlab

- É um software científico para computação numérica.



Matlab com cuda

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

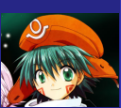
Cuda

Matlab com
cuda

Bibliografia

Oque e o Matlab

- É um software científico para computação numérica.
- Possui uma grande comunidade.



Matlab com cuda

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

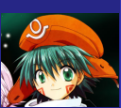
Cuda

Matlab com
cuda

Bibliografia

Oque e o Matlab

- É um software científico para computação numérica.
- Possui uma grande comunidade.
- Nele possível criar rotinas em C ou Fortran(estas rotinas são mais rápidas que as criadas na linguagem nativa do matlab).



Matlab com cuda

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

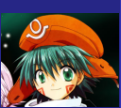
Cuda

Matlab com
cuda

Bibliografia

Oque e o Matlab

- E um software científico para computação numérica.
- Possui uma grande comunidade.
- Nele possível criar rotinas em C ou Fortran(estas rotinas são mais rápidas que as criadas na linguagem nativa do matlab).
- Possui um plug-in, disponibiliza-do pela nvidia, pra se fazer a ligação entre ele e o cuda.



Matlab com cuda

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

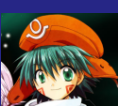
Cuda

Matlab com
cuda

Bibliografia

No próximo slide um exemplo de código em C, para Matlab (sem usar CUDA), que pega o elemento de um vetor eleva cada elemento ao quadrado.

O nome do arquivo fonte é `square_matlab.c`



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

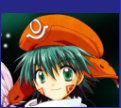
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

```
1  #include "mex.h"
2  void mexFunction(int nlhs, mxArray *plhs[],int nrhs, const mxArray *prhs[])
3  {
4      int i, j, m, n;
5      double *data1, *data2;
6      /* Checa se o numero de argumentos passados esta correto */
7      if (nrhs != 1 || nlhs !=1 )
8          mexErrMsgTxt("0 numero de argumentos de entrada ou de saida esta incorreto");
9
10     else{
11         /* Extrai as dimensoes da variavel de entrada*/
12         m = mxGetM(prhs[0]);
13         n = mxGetN(prhs[0]);
14         /* Cria uma variavel de saida */
15         plhs[0] = mxCreateDoubleMatrix(m, n, mxREAL);
16         /* Cria-se um ponteiro para a variavel de entrada */
17         data1 = mxGetPr(prhs[0]);
18         /* Cria-se um ponteiro para variavel de saida */
19         data2 = mxGetPr(plhs[0]);
20         /* Faz a conta, e coloca o resultado na variavel de saida*/
21         for (j = 0; j < m*n; j++){
22             data2[j] = data1[j] * data1[j];
23         }
24     }
25 }
```



Compilando o arquivo e executando a rotina

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

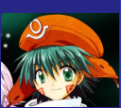
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

- 1 Para se compilar este arquivo entre no matlab, aponte o path do matlab para o diretorio onde esta o arquivo é digite `mex square_matlab.c`



Compilando o arquivo e executando a rotina

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

Matlab com
cuda

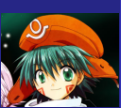
Bibliografia

- 1 Para se compilar este arquivo entre no matlab, aponte o path do matlab para o diretorio onde esta o arquivo é digite `mex square_matlab.c`
- 2 Para se fazer a link do arquivo binário, resultante da compilação acima, com o matlab digite:

`which square_matlab square_matlab.mexw32` - Para Windows

`which square_matlab square_matlab.mexglx` - Para linux 32 bits

`which square_matlab square_matlab.mexa64` - Para linux 64 bits



Compilando o arquivo e executando a rotina

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

❶ Para se compilar este arquivo entre no matlab, aponte o path do matlab para o diretorio onde esta o arquivo é digite `mex square_matlab.c`

❷ Para se fazer a link do arquivo binário, resultante da compilação acima, com o matlab digite:

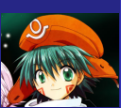
`which square_matlab square_matlab.mexw32` - Para Windows

`which square_matlab square_matlab.mexglx` - Para linux 32 bits

`which square_matlab square_matlab.mexa64` - Para linux 64 bits

❸ Um exemplo de como executar a rotina no matlab:

`saida=square_matlab([1 2 3 4])`



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

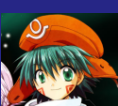
Cuda

Matlab com
cuda

Bibliografia

No próximo slide um exemplo de código em Cuda, para Matlab, que pega os elementos de um vetor e eleva cada elemento ao quadrado.

O nome do arquivo fonte é `square_matlab.cu`



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

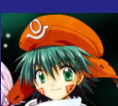
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

```
1  #include "cuda.h"
2  #include "mex.h"
3  /* Kernel to square elements of the array on the GPU */
4  __global__ void square_elements(float* in, float* out, int N){
5      int idx = blockIdx.x*blockDim.x+threadIdx.x;
6      if ( idx < N) out[idx]=in[idx]*in[idx];
7  }
8  /* Gateway function */
9  void mexFunction(int nlhs, mxArray *plhs[],int nrhs, const mxArray *prhs[]){
10     int j, m, n;
11     double *data1, *data2;
12     float *data1f, *data2f;
13     float *data1f_gpu, *data2f_gpu;
14     if (nrhs != 1 || nlhs != 1){
15         mexErrMsgTxt("0 numero de argumentos de entrada ou saida passados para a funÃ
16     }
17     else{
18         /* Find the dimensions of the data */
19         m = mxGetM(prhs[0]);
20         n = mxGetN(prhs[0]);
21         /* Create an mxArray for the output data */
22         plhs[0] = mxCreateDoubleMatrix(m, n, mxREAL);
23         /* Create an input and output data array on the GPU*/
24         cudaMalloc( (void **) &data1f_gpu, sizeof(float)*m*n);
25         cudaMalloc( (void **) &data2f_gpu, sizeof(float)*m*n);
26         /* Retrieve the input data */
27         data1 = mxGetPr(prhs[0]);
28         /* Check if the input array is single or double precision */
29         /* The input array is in double precision, it needs to be converted t floats before bei
30         data1f = (float *) mxMalloc(sizeof(float)*m*n);
31         for (j = 0; j < m*n; j++){
```



Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

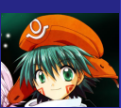
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

```
32         data1f[j] = (float) data1[j];
33     }
34     cudaMemcpy( data1f_gpu, data1f, sizeof(float)*n*m, cudaMemcpyHostToDevice);
35     data2f = (float *) mxMalloc(sizeof(float)*m*n);
36     /* Compute execution configuration using 128 threads per block */
37     int dimBlock=128;
38     int dimGrid=((m*n)/dimBlock);
39     if ( (n*m) % 128 !=0 ) dimGrid+=1;
40     /* Call function on GPU */
41     square_elements<<<dimGrid,dimBlock>>>(data1f_gpu, data2f_gpu, n*m);
42     /* Copy result back to host */
43     cudaMemcpy( data2f, data2f_gpu, sizeof(float)*n*m, cudaMemcpyDeviceToHost);
44     /* Create a pointer to the output data */
45     data2 = mxGetPr(plhs[0]);
46     /* Convert from single to double before returning */
47     for (j = 0; j < m*n; j++){
48         data2[j] = (double) data2f[j];
49     }
50     /* Clean-up memory on device and host */
51     mxFree(data1f);
52     mxFree(data2f);
53     cudaFree(data1f_gpu);
54     cudaFree(data2f_gpu);
55 }
56 }
57
```



Compilando o arquivo e executando a rotina

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

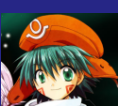
Matlab com
cuda

Bibliografia

- 1 No shell do SO, dentro da pasta onde esta o arquivo fonte, digite o seguinte comando:

```
$diretorio_do_plug-in/nvmex -f $diretorio_do_plug-in/nvopts.sh square_matlab.cu -L
```

```
$diretorio_do_cuda/lib -lcudart
```



Compilando o arquivo e executando a rotina

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

- 1 No shell do SO, dentro da pasta onde esta o arquivo fonte, digite o seguinte comando:

```
$diretorio_do_plug-in/nvmex -f $diretorio_do_plug-in/nvopts.sh square_matlab.cu -L
```

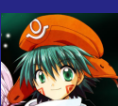
```
$diretorio_do_cuda/lib -lcudart
```

- 2 Para se fazer a link do arquivo binário, resultante da compilação acima, entre no matlab, aponte o path do matlab para o diretorio onde esta o arquivo é digite:

`which square_matlab square_matlab.mexw32` - Para Windows

`which square_matlab square_matlab.mexglx` - Para linux 32 bits

`which square_matlab square_matlab.mexa64` - Para linux 64 bits



Compilando o arquivo e executando a rotina

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

- 1 No shell do SO, dentro da pasta onde esta o arquivo fonte, digite o seguinte comando:

```
$diretorio_do_plug-in/nvmex -f $diretorio_do_plug-in/nvopts.sh square_matlab.cu -L
```

```
$diretorio_do_cuda/lib -lcudart
```

- 2 Para se fazer a link do arquivo binário, resultante da compilação acima, entre no matlab, aponte o path do matlab para o diretorio onde esta o arquivo é digite:

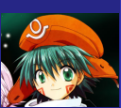
`which square_matlab square_matlab.mexw32` - Para Windows

`which square_matlab square_matlab.mexglx` - Para linux 32 bits

`which square_matlab square_matlab.mexa64` - Para linux 64 bits

- 3 Um exemplo de como executar a rotina no matlab:

```
saida=square_matlab([1 2 3 4])
```



Sumário

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

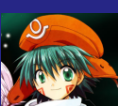
Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

- 1 Introdução
- 2 Comparação entre GPUs e CPUs
- 3 Arquitetura de GPUs
- 4 Cuda
- 5 Matlab com cuda
- 6 Bibliografia**



Bibliografia

Utilização da
unidade
processamento
gráfico para
propósito
geral
(GPGPU)

Rafael Guedes
Lang
Anderson
Gonçalves
Marco

Introdução

Comparação
entre GPUs e
CPUs

Arquitetura de
GPUs

Cuda

Matlab com
cuda

Bibliografia

www.nvidia.com/cuda

<http://forums.nvidia.com/index.php?showforum=62>

www.gpgpu.org

www.clubedohardware.com.br

www.mathematik.uni-dortmund.de/~goeddeke/arcs2008/C1_CUDA.pdf

home.in.tum.de/~heinecke/docs/heinecke_fa2007_slides.pdf

NVIDIA Cuda Programming Guide