Programação Paralela em GPU

Conceitos e paradigmas

Rogério Perino de O. Neves
Universidade Federal do ABC

rogerio.neves@ufabc.edu.br

- Informações gerais
- Conceitos de computação paralela
- Por que GPU?
- Programando em GPUs

UFABC: Proposta

- Proposta por Luiz Bevilacqua (2º reitor), fundada julho de 2005
- Corpo docente unicamente de doutores (100%)
- Projeto pedagógico interdisciplinar, divisão em 3 centros:
  - Centro de Matemática, Computação e Cognição (CMCC)
  - Centro de Ciências Naturais e Humanas (CCNH)
  - Centro de Engenharia e Ciências Sociais (CECS)
- 3 campus previstos originalmente, novos em estudo/negociação:
  - Em funcionamento: Santo André, São Bernardo do Campo
  - Em projeto: São Caetano, Mauá

UFBAC: Dados

- 1º lugar no Ranking SCImago entre as brasileiras nos quesitos:
  - Excelência em Pesquisa
  - Publicações de alta qualidade
  - Impacto normalizado das suas publicações
- IGC do MEC
  - Melhor do Estado
  - 1ª do Brasil em cursos de graduação
- Ranking Universitário Folha 2013
  - 1º lugar em “Internacionalização”
- A única brasileira com fator de impacto acima da média mundial (SCImago)

Rogério Neves: Formação

- Formado pelo IFSC-USP em Física computacional (2000)
- Mestrado pela POLI-USP em Engenharia de Sistemas (2003)
- Doutor pela YNU* em Engenharia de Sistemas (2006)
- Na UFABC desde Março/2009

* Yokohama National University, Yokohama, Japão

Rogério Neves: Áreas de trabalho

- Grupos de pesquisa
  - Grupo de Computação Paralela e Distribuída
  - Grupo de Visão Computacional

- Trabalhos publicados nas áreas de:
  - Robótica
  - Reconhecimento de padrões
  - Sistemas de Controle
  - Engenharia naval
  - Vida Artificial

- Em desenvolvimento
  - Computação quântica

- Informações gerais
- Conceitos de computação paralela
- Por que GPU?
- Programando em GPUs

Conceitos de Computação paralela

Conceitos fundamentais:

- Threads
- Gargalos
- Granularidade (fina, grossa, embaraçosamente paralelo)
- Paralelismo (alto/baixo)
- (In)dependência entre nós
- Comunicação entre nós
- Latência na comunicação
- Speedup (aceleração)
- Modelos de memória

Tipos de paralelismo

- Em nível de bits (4-bits, 8-bits... 64-bits)
- Em nível de instrução (pipelines)
- Multithread (várias linhas de execução)
- Multicore (vários processadores)

- Taxonomia de Flynn:

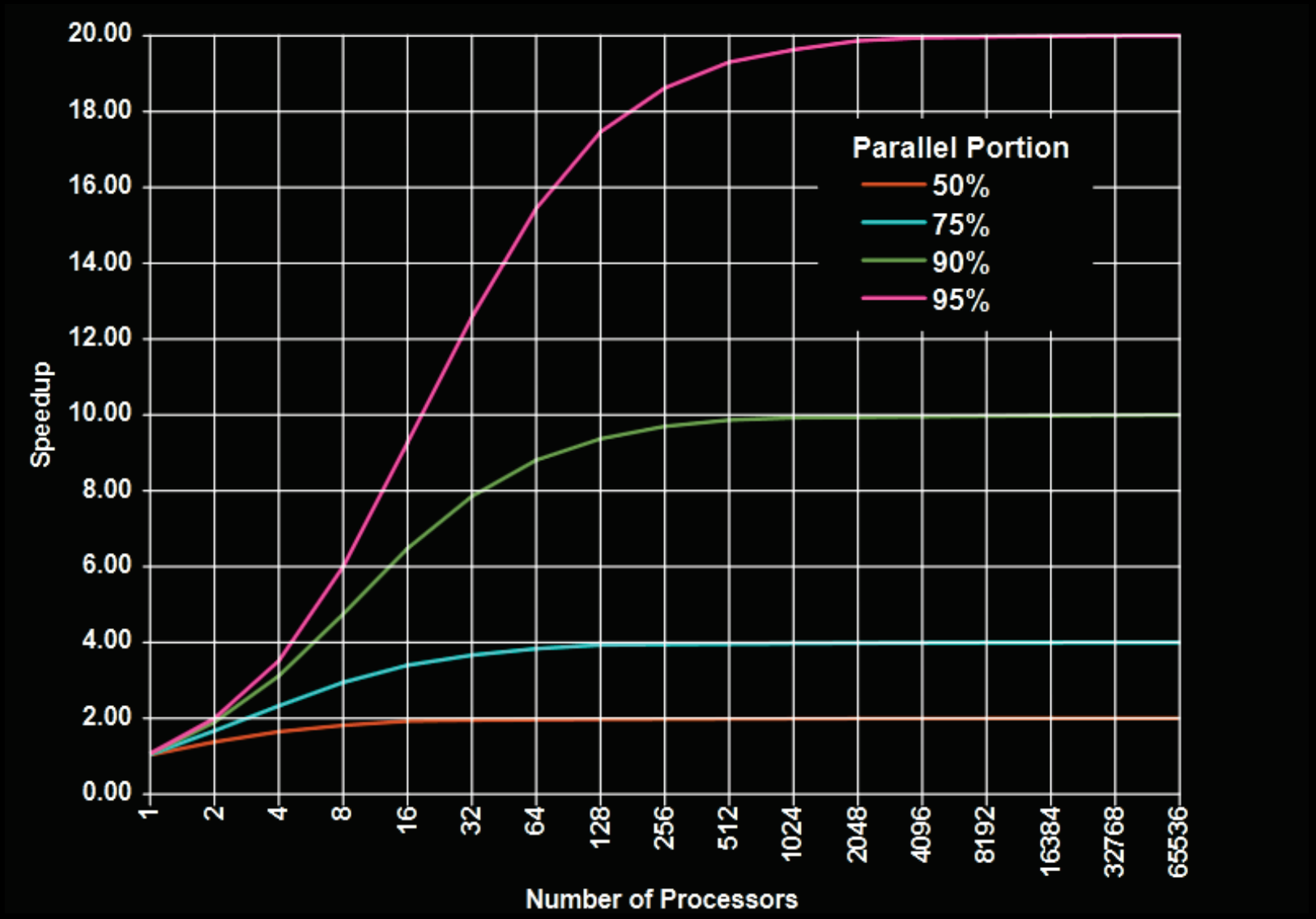
SISD, SIMD, MIMD, MISD

Tipos de paralelismo

- Em nível de bits (4-bits, 8-bits... 64-bits)
- Em nível de instrução (pipelines)
- Multithread (várias linhas de execução)
- Multicore (vários processadores)

SISD, SIMD, MIMD, MISD

Lei de Amdahl



Problemas em programação paralela

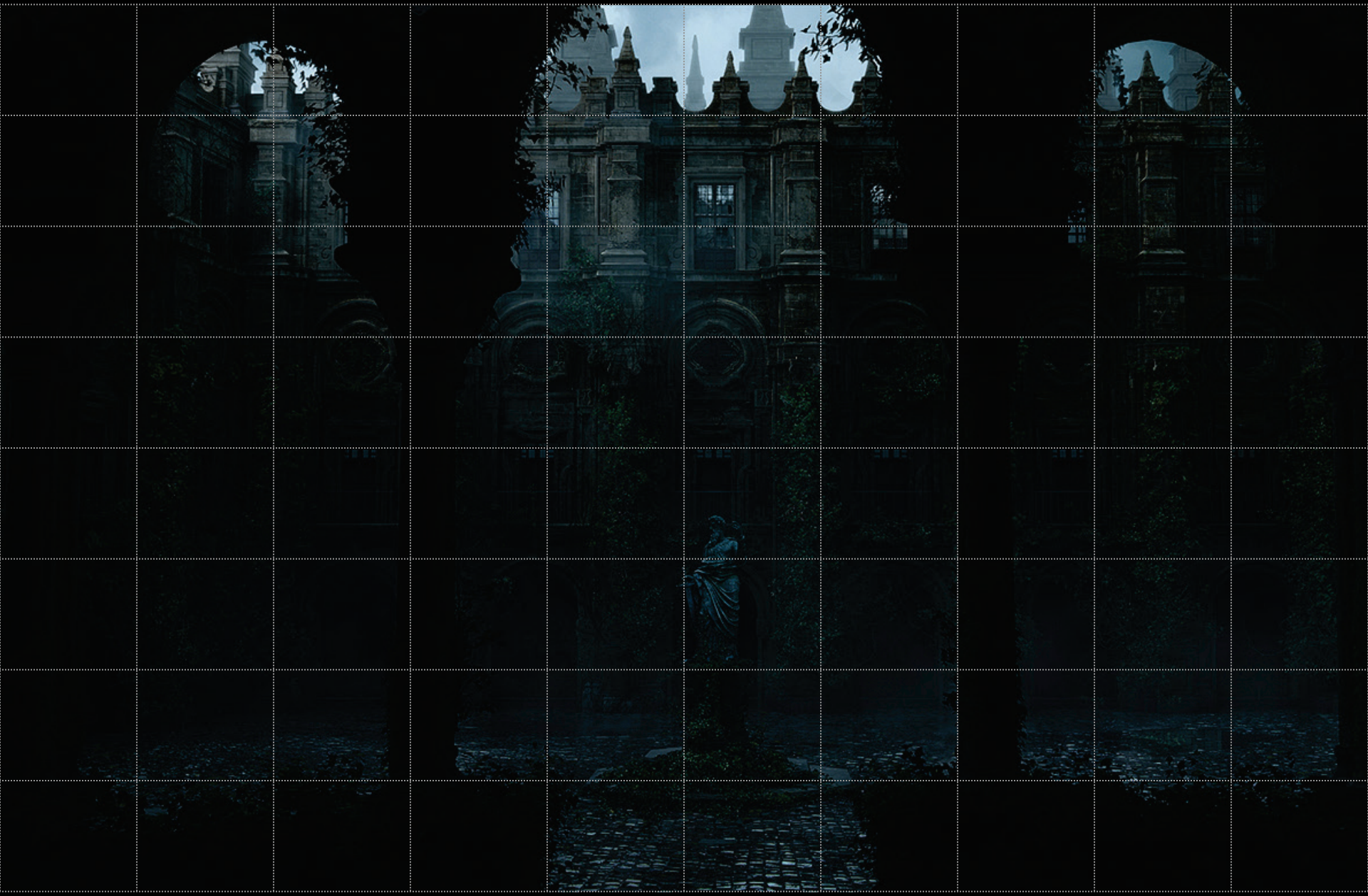
- Race conditions
- Mutual exclusion
- Synchronization
- Parallel slowdown

Paralelismo

- Qual o mais potente?

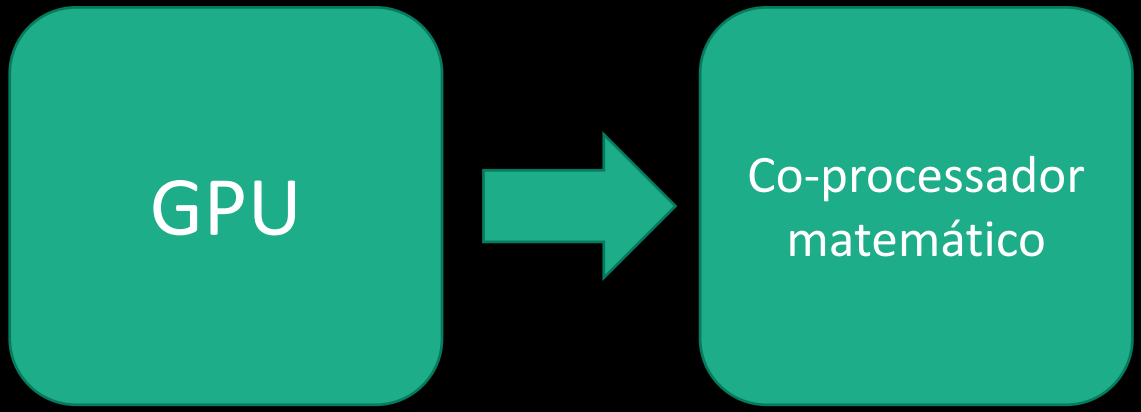
a. 1 touro
b. 1000 frangos

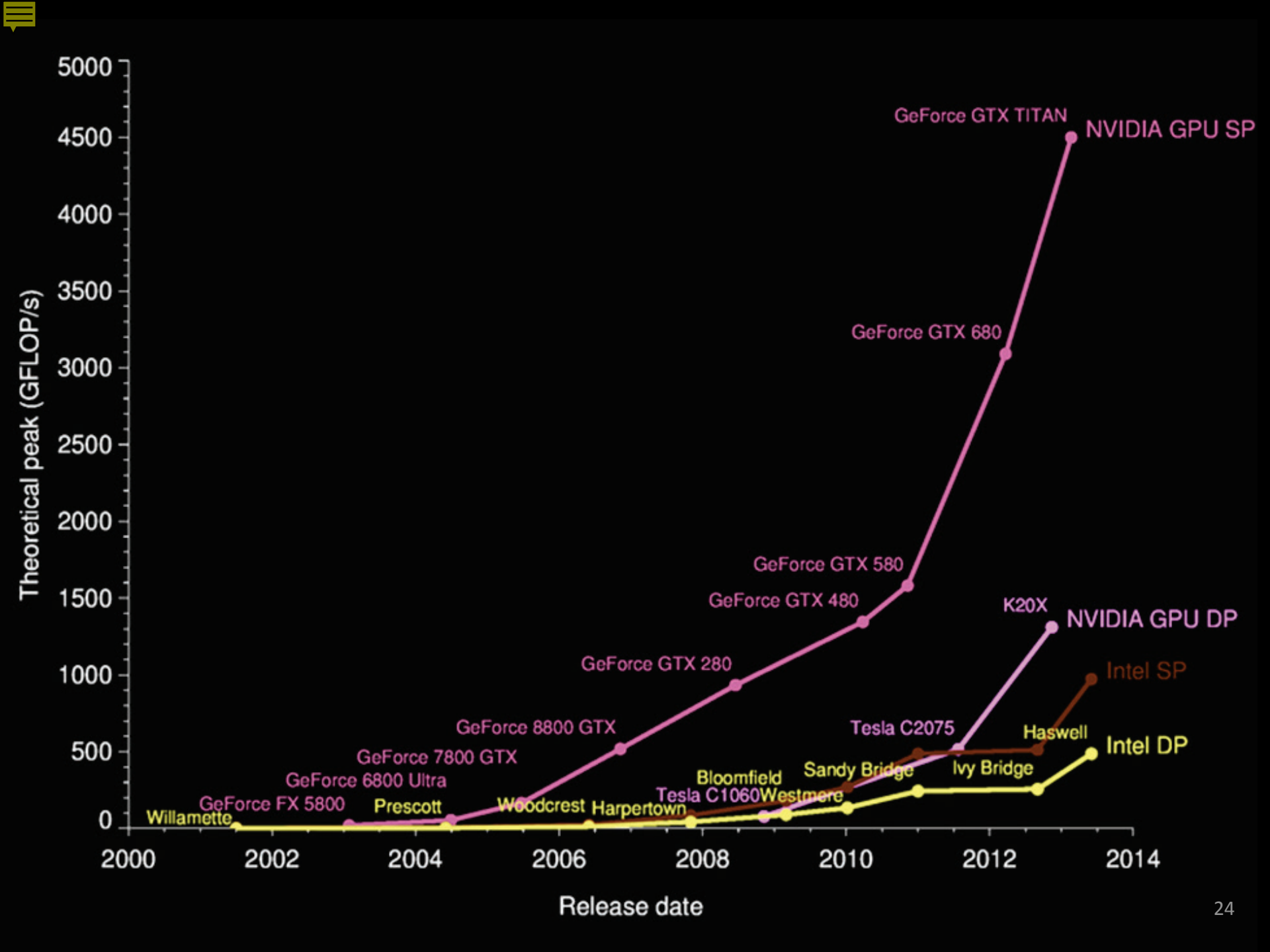
- Informações gerais
- Conceitos de computação paralela
- Por que GPU?
- Programando em GPUs



GPGPU

- General-purpose computing (GP) on graphics processing units (GPU) :
  - Utiliza o poder não aproveitado das placas gráficas
  - Permitir o uso irrestrito de rotinas customizadas




5000
4500
4000
3500
3000
2500
2000
1500
1000
500
0
Theoretical peak (GFLOP/s)
2000
2002
2004
2006
2008
2010
2012
2014
Release date
GeForce GTX TITAN
NVIDIA GPU SP
GeForce GTX 680
GeForce GTX 580
GeForce GTX 480
GeForce GTX 280
GeForce 8800 GTX
GeForce 7800 GTX
GeForce 6800 Ultra
GeForce FX 5800
Willamette
Prescott
Woodcrest
Harpertown
Bloomfield
Tesla C1060
Westmere
Sandy Bridge
Ivy Bridge
Haswell
K20X
NVIDIA GPU DP
Tesla C2075
Intel SP
Intel DP

Silicon Graphics Altix 4700 Bachianas

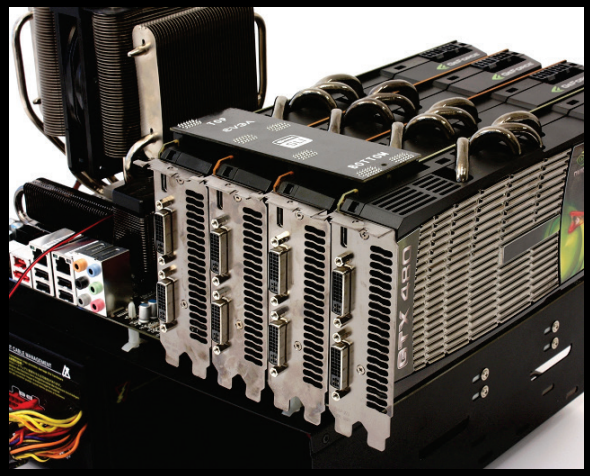
- Características
  - Adquirido em Agosto de 2007
  - 136 processadores (cores)
    Itanium 2 (64 bits)
  - 272 GB de memória RAM
  - 30 TB de em disco
  - Linux (SUSE) e Enterprise Server 10
  - 2,4 TFLOP/s de desempenho de pico
  - R$ 2 milhões

- Aplicações:
  - Quantum Dots
  - Fluid dynamics
  - Outras: Política de Filas

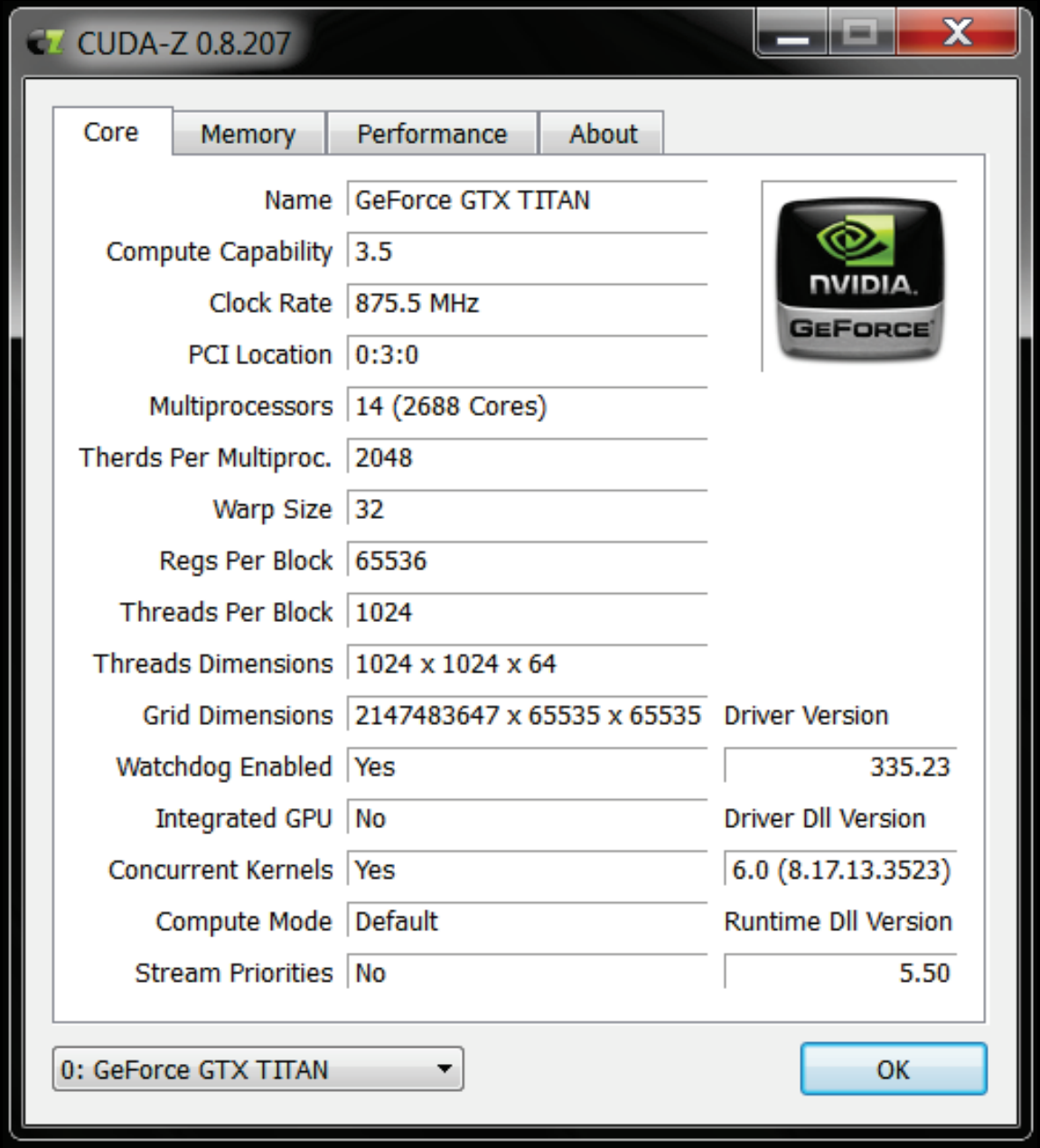
Custo ponderado = 833,3 R$/GigaFLOP

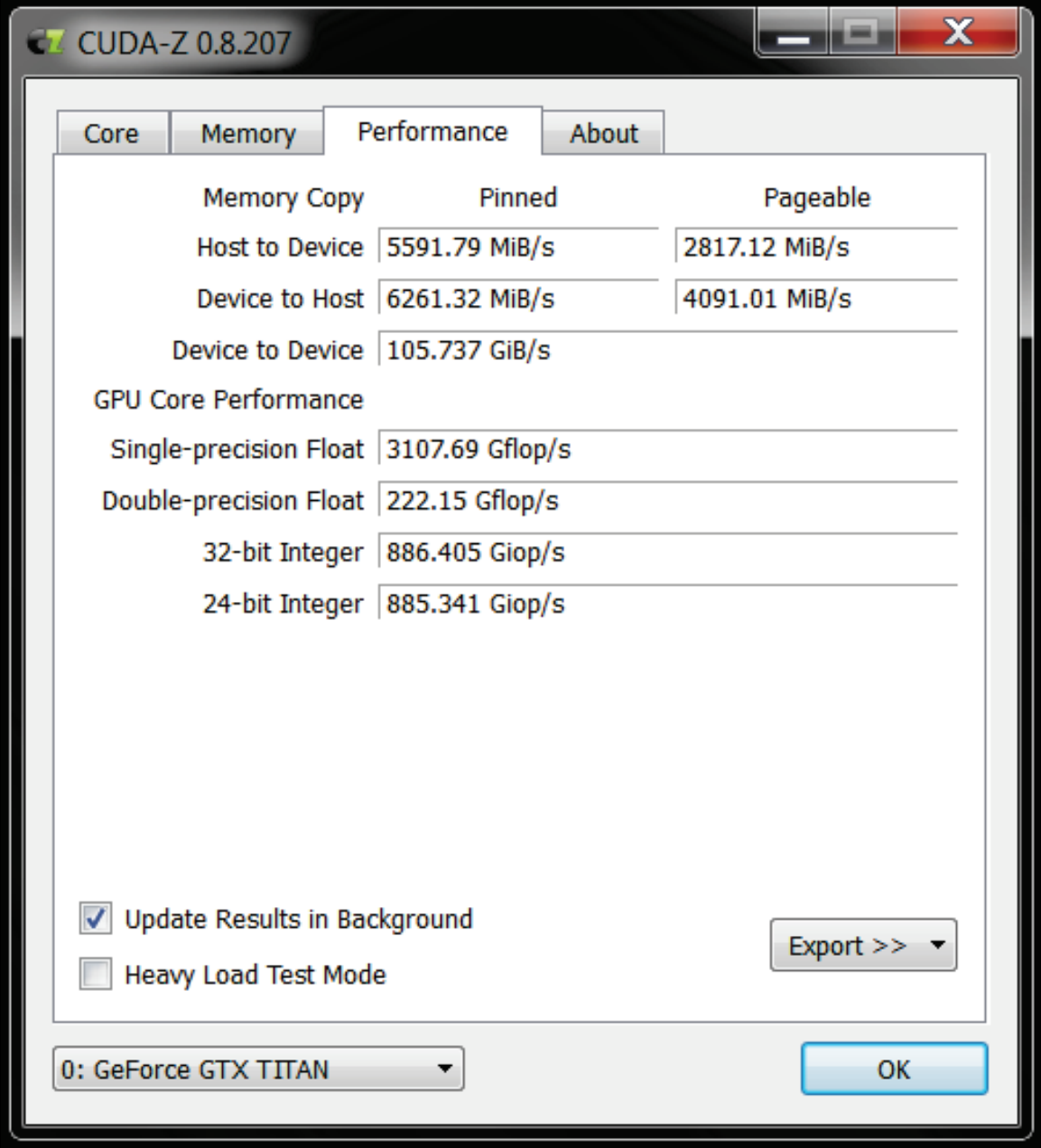
GPU Supercomputer

- Características
  - Intel Core 2 duo
  - 16GB de memória RAM
  - 8 TB de disco (4x2 RAID)
  - 4 x GeForce GTX 480
    - 1792 cores (448 X 4)
    - 5 GB de memória (1280 x 4)
    - 5,4 TFLOP/S desempenho teórico (1,35 x 4)
    - R$ 16.840,00

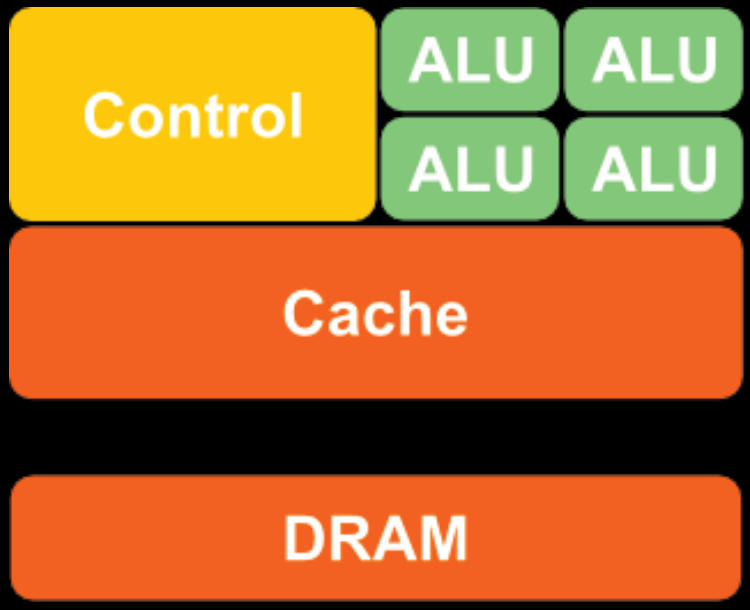


Custo ponderado = 3,11 R$/GigaFLOP

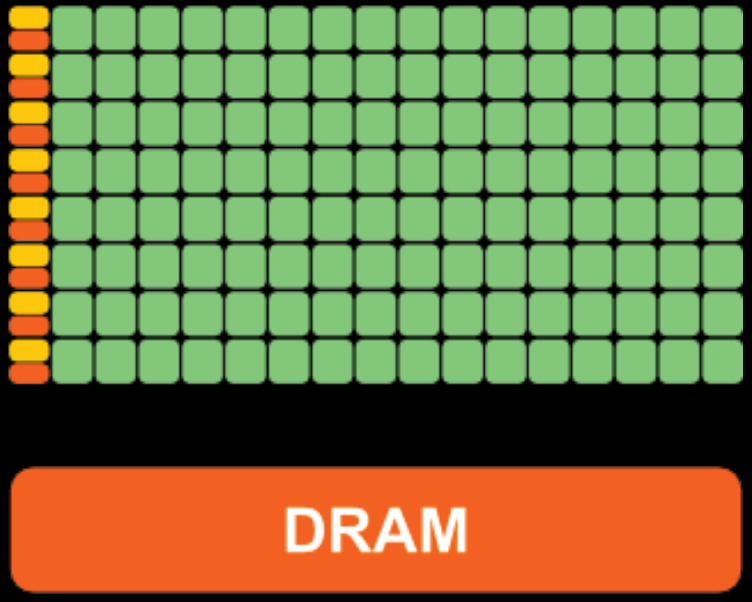

CUDA-Z 0.8.207
Core
Memory
Performance
About
Name
GeForce GTX TITAN
Compute Capability
3.5
Clock Rate
875.5 MHz
PCI Location
0:3:0
Multiprocessors
14 (2688 Cores)
Therds Per Multiproc.
2048
Warp Size
32
Regs Per Block
65536
Threads Per Block
1024
Threads Dimensions
1024 x 1024 x 64
Grid Dimensions
2147483647 x 65535 x 65535
Watchdog Enabled
Yes
Integrated GPU
No
Concurrent Kernels
Yes
Compute Mode
Default
Stream Priorities
No
Driver Version
335.23
Driver Dll Version
6.0 (8.17.13.3523)
Runtime Dll Version
5.50
NVIDIA
GeForce
0: GeForce GTX TITAN
OK


CUDA-Z 0.8.207
Core
Memory
Performance
About
Memory Copy
Pinned
Pageable
Host to Device
5591.79 MiB/s
2817.12 MiB/s
Device to Host
6261.32 MiB/s
4091.01 MiB/s
Device to Device
105.737 GiB/s
GPU Core Performance
Single-precision Float
3107.69 Gflop/s
Double-precision Float
222.15 Gflop/s
32-bit Integer
886.405 Giop/s
24-bit Integer
885.341 Giop/s
Update Results in Background
Heavy Load Test Mode
Export >>
0: GeForce GTX TITAN
OK

CPU x GPU

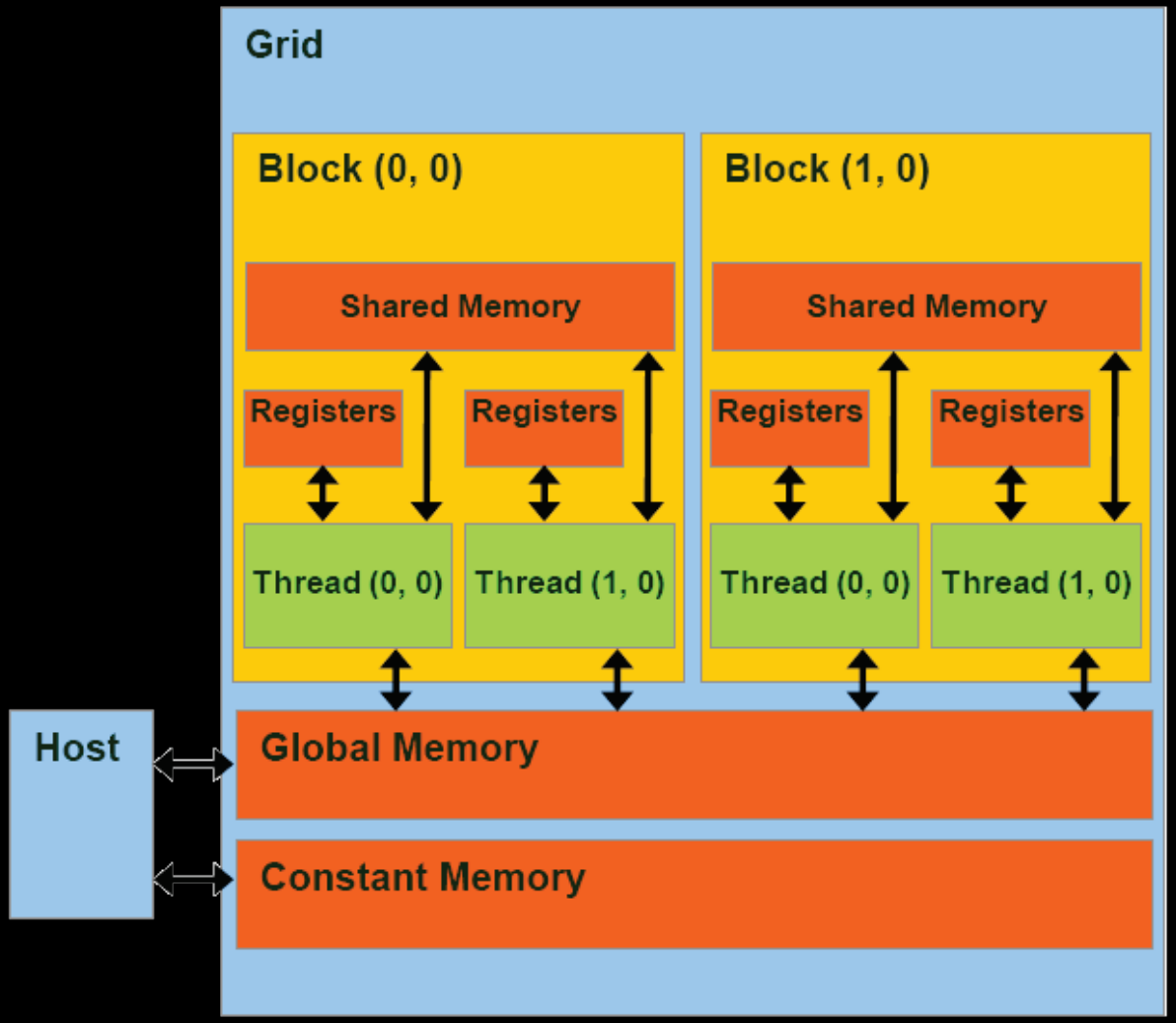


CPU



GPU

Arquitetura GPU



- Informações gerais
- Conceitos de computação paralela
- Por que GPU?
- Programando em GPUs

Prográmação em GPU

- CUDA (C, C++, Fortran)
- OpenMP
- Matlab
  - R2010b ou posterior
  - GPU com compute capability 1.3 ou superior

GPGPU com CUDA C

C Padrão

```
void saxpy_serial(int n,
                  float a,
                  float *x,
                  float *y)
{
  for (int i = 0; i < n; ++i)
    y[i] = a*x[i] + y[i];
}

// Perform SAXPY on 1M elements
saxpy_serial(4096*256, 2.0, x, y);
```

C Paralelo

```
__global__
void saxpy_parallel(int n,
                    float a,
                    float *x,
                    float *y)
{
  int i = blockIdx.x*blockDim.x +
          threadIdx.x;
  if (i < n) y[i] = a*x[i] + y[i];
}

// Perform SAXPY on 1M elements
saxpy_parallel<<<4096, 256>>>(n,2.0,x,y);
```

http://developer.nvidia.com/cuda-toolkit

CUDA Fortran

Fortran90

```
module stencil
  integer, parameter:: radius = 3
  contains
  subroutine applyStencil1D( N, in, out )
    integer:: i, j
    integer:: N
    real:: in( N ), out( N )

    !loop over interior elements
    do i = radius+1, N-radius
      out( i ) = 0
      do j = -radius, radius
        out( i ) = out( i ) + in( i + j )
      end do
    end do
  end subroutine
end module
```

CUDA Fortran

```
module stencil
  integer, parameter:: radius = 3
  contains
  attributes( global ) subroutine applyStencil1D( N, in, out )
    integer:: i, j
    integer, value :: N
    real:: in( N ), out( N )
    !compute element index
    i = threadIdx%x + (blockIdx%x-1) * blockDim%x
    if ( i .gt. radius .and. i .le. N-radius) then
      out( i ) = 0
      do j = -radius, radius
        out( i ) = out( i ) + in( i + j )
      end do
    end if
  end subroutine
end module
```

Exemplo CUDA-C: Primos

```
1  #include <stdio.h>
2  #include <cuda.h>
3  #include <time.h>
4  __global__ void gpu (float *num){
5          int id = blockIdx.x * 256 +threadIdx.x;
6          float n=num[id];
7          float fim = sqrt(n);
8          for (float i = 3; i <= fim; i = i + 2) {
9                  if ((remainder(n,i))==0) {
10                         num[id] = 0;
11                         return;
12 }       }       }
13 void cpu (float * num, int s) {
14 //      printf("\n%d numbers",s);
15         for (int j = 0; j<s; j++) {
16                 float fim = num[j]-2;
17                 for (float i = 3; i <= fim; i = i + 2) {
18                         if ((remainder(num[j],i))==0) {
19                                 num[j] = 0;
20                                 return;
21 }       }       }       }
```

Bibliotecas CUDA

- CUBLAS - basic linear algebra
- CUFFT - fourier transforms
- CUDPP - data parallel primitives
- CURAND - random number generation

GPGPU com Matlab

- Biblioteca JACKET (2007-2012)
- Suporte nativo a partir da versão R2010b
- Nº de funções e toolboxes aumenta a cada versão*
- Maneira mais simples de começar a utilizar a GPU

* www.mathworks.com/help/distcomp/release-notes.html

Meu sistema tem alguma GPU?

```
>> n=gpuDeviceCount

n =

     2

>> d1=gpuDevice(1)

d1 =

  CUDADevice with properties:

                      Name: 'GeForce GTX TITAN'
                     Index: 1...
```

```
>> d1 = gpuDevice(1)

  CUDADevice with properties:

                      Name: 'GeForce GTX TITAN‘ (1)
       ComputeCapability: '3.5'
          SupportsDouble: 1
           DriverVersion: 6
          ToolkitVersion: 6
       MaxThreadsPerBlock: 1024
         MaxShmemPerBlock: 49152
       MaxThreadBlockSize: [1024 1024 64]
              MaxGridSize: [2.1475e+09 65535 65535]
              TotalMemory: 6.4425e+09
               FreeMemory: 5.9389e+09
     MultiprocessorCount: 14
            ClockRateKHz: 875500
```

```
>> d1=gpuDevice(1)

d1 =

  CUDADevice with properties:

                      Name: 'GeForce GTX TITAN'
                     Index: 1
         ComputeCapability: '3.5'
            SupportsDouble: 1
             DriverVersion: 6
            ToolkitVersion: 5
        MaxThreadsPerBlock: 1024
          MaxShmemPerBlock: 49152
        MaxThreadBlockSize: [1024 1024 64]
               MaxGridSize: [2.1475e+09 65535 65535]
                 SIMDWidth: 32
               TotalMemory: 6.4425e+09
                FreeMemory: 5.9386e+09
       MultiprocessorCount: 14
              ClockRateKHz: 875500
               ComputeMode: 'Default'
      GPUOverlapsTransfers: 1
    KernelExecutionTimeout: 1
          CanMapHostMemory: 1
           DeviceSupported: 1
            DeviceSelected: 1
```

```
>> d2=gpuDevice(2)

d2 =

  CUDADevice with properties:

                      Name: 'GeForce GTX 570'
                     Index: 2
         ComputeCapability: '2.0'
            SupportsDouble: 1
             DriverVersion: 6
            ToolkitVersion: 5
        MaxThreadsPerBlock: 1024
          MaxShmemPerBlock: 49152
        MaxThreadBlockSize: [1024 1024 64]
               MaxGridSize: [65535 65535 65535]
                 SIMDWidth: 32
               TotalMemory: 1.3422e+09
                FreeMemory: 1.2133e+09
       MultiprocessorCount: 15
              ClockRateKHz: 1464000
               ComputeMode: 'Default'
      GPUOverlapsTransfers: 1
    KernelExecutionTimeout: 1
          CanMapHostMemory: 1
           DeviceSupported: 1
            DeviceSelected: 1
```

>> methods('gpuArray')

```
Methods for class gpuArray:

abs                 divergence          ipermute            real
acos                dot                 isa                 reallog
acosh               double              isempty             realpow
acot                eig                 isequal             realsqrt
acoth               end                 isequaln            reducepatch
acsc                eps                 isequalwithequalnans reducevolume
acsch               eq                  isfinite            rem
all                 erf                 isfloat             repmat
and                 erfc                isinf               reshape
any                 erfcinv             isinteger           ribbon
applylut            erfcx               islogical           rose
area                erfinv              ismember            round
arrayfun            errorbar            isnan               scatter3
asec                existsOnGPU         isnumeric           sec
asech               exp                 isocaps             sech
asin                expm1               isocolors           semilogx
asinh               ezcontour           isonormals          semilogy
atan                ezcontourf          isosurface          shiftdim
atan2               ezgraph3            isreal              shrinkfaces
atanh               ezmesh              issorted            sign
bar                 ezmeshc             issparse            sin
bar3                ezplot              ldivide             single
```

Exemplo Matlab: FFT

- Fast Fourier Transform com CPU

```
>> clear
>> M=rand(1,100000000,'single');
>> tic; N=fft(M); toc
Elapsed time is 2.650846 seconds.
```

- Fast Fourier Transform com GPU

```
>> reset(gpuDevice())
>> GM=gpuArray.rand(1,100000000,'single');
>> tic; GN=fft(GM); toc
Elapsed time is 0.097268 seconds.

>> N=gather(GN);
```

SU = 27,3

Exemplo Matlab: FFT

- Fast Fourier Transform com CPU

```
>> M=rand(1,100000000,'single');

>> tic; T=fft(M); toc
Elapsed time is 2.704970 seconds.

>> reset(gpuDevice())

>> GM=gpuArray(M);

>> tic; GT=fft(GM); toc
Elapsed time is 0.065170 seconds.

>> T=gather(GT);
```

SU = 41,5

Exemplo Matlab: Matriz inversa

- CPU

```
>> clear
>> X=rand(10000,10000)*20-10;
>> tic; Y=inv(X); toc
Elapsed time is 69.501617 seconds.
```

- GPU

```
>> reset(gpuDevice())
>> GX=gpuArray(X);
>> tic; GY=inv(GX); toc
Elapsed time is 15.056601 seconds.

>> Y=gather(GY);
```

SU = 4,6

Exemplo Matlab: Matriz inversa

- Matriz inversa, precisão simples

```
>> clear
>> reset(gpuDevice())

>> X=rand(10000,10000, 'single')*20-10;
>> tic; IX=inv(X); toc
Warning: Matrix is close to singular or badly
scaled. Results may be inaccurate. RCOND =
6.044021e-08.
Elapsed time is 32.429261 seconds.

>> GX=gpuArray(X);
>> tic; GIX=inv(GX); toc
Elapsed time is 2.305418 seconds.
```

SU = 14

Exemplo Matlab: Determinante

- Cálculo do determinante na CPU e na GPU

```
>> clear
>> reset(gpuDevice())

>> X=rand(10000,10000, 'single')*20-10;
>> tic; determinante=det(X); toc
Elapsed time is 10.216976 seconds.

>> GX=gpuArray(X);
>> tic; Gdeterminante=det(GX); toc
Elapsed time is 0.904586 seconds.
```

SU = 11,4

Exemplo Matlab: Determinante

- Cálculo do determinante na CPU e na GPU

```
>> clear
>> reset(gpuDevice())

>> X=rand(10000,10000, 'double')*20-10;
>> tic; determinante=det(X); toc
Elapsed time is 20.262518 seconds.

>> GX=gpuArray(X);
>> tic; Gdeterminante=det(GX); toc
Elapsed time is 4.096936 seconds.
```

SU < 5

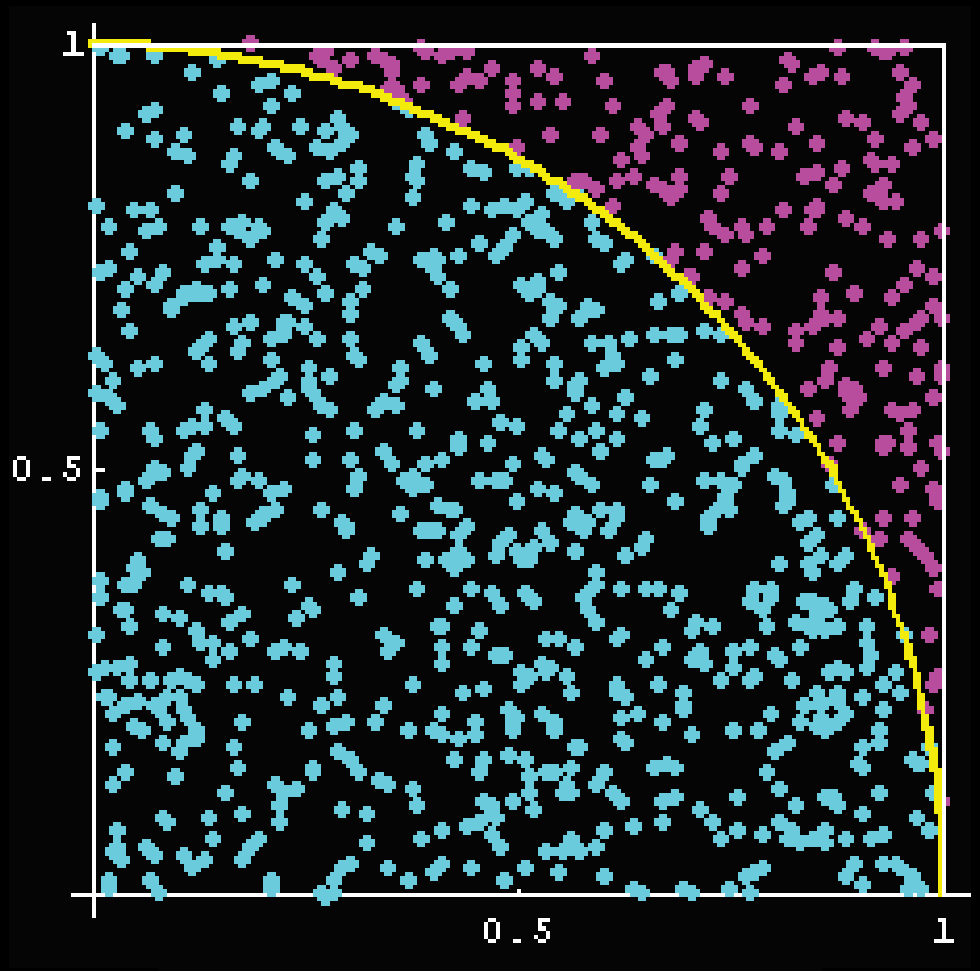
Exemplo: Estimação de π

- Método de Monte Carlo
  - Pares sorteados (x, y) entre 0 e 1
  - n é o nº total de pontos (dentro do quadrado)
  - m é o nº de pontos dentro do circulo

$$\rho = \frac{\text{Area of Circle}}{\text{Area of Square}} = \frac{\pi r^2}{(2r)^2} = \frac{\pi}{4} = \frac{3.1415926535897932}{4} = 0.78$$

$$\pi \approx \rho * 4 = \frac{m}{n} * 4$$

Exemplo: Estimação de π



$$\pi \approx \rho * 4 = \frac{m}{n} * 4$$

Exemplo: Estimação de π

- Em CPU

```
>> clear

>> n=4E8;

>> x=rand(1,n);
>> y=rand(1,n);

>> tic; m = sum(sqrt(x.^2+y.^2)<=1); toc
Elapsed time is 2.064938 seconds.

>> pimc=4*m/n;
>> disp([pi; pimc])
   3.141592653589793
   3.141577000000000
```

Exemplo: Estimação de π

- CPU

```
Clear
format long

n=3E8;
tic;
x=rand(1,n);
y=rand(1,n);
m = sum(sqrt(x.^2+y.^2)<=1);
toc

pimc=4*m/n;
disp([pi; pimc])
```

- GPU

```
Clear
format long
reset(gpuDevice())
n=3E8;
tic;
x=gpuArray.rand(1,n,'single');
y=gpuArray.rand(1,n,'single');
gm = sum(sqrt(x.^2+y.^2)<=1);
toc
m=gather(gm);
pimc=4*m/n;
disp([pi; pimc])
```

Exemplo: Estimação de π

- Em GPU

```
>> pimccpu
Elapsed time is 10.578446 seconds.
   3.141592653589793
   3.141546373333334

>> pimcgpu
Elapsed time is 1.389589 seconds.
 3.141592653589793
 3.141595826666667

>>
```

SU = 7,6

Exemplo: copiar/criar dados

- Copiar dados para a GPU

```
G=gpuArray(D)
G=gpuArray(single(zeros(10,10))
```

- Criar dados direto na GPU

```
G=gpuArray.rand(10,10, 'single')
G=gpuArray.eye(10,'single')
G=gpuArray.zeros(10,10))
```

- Copiar dados da GPU

```
D=gather(G)
```

Copiar dados: CUDA

```
void *d_array = cudaMalloc(sizeof(void*) * N);
// Copy to device Memory
cudaMemcpy(d_array, h_array, sizeof(void*) * N,
      cudaMemcpyHostToDevice);

multi_array_kernel<1,1>(N_ARRAYS, d_array);

cudaThreadSynchronize();

cudaMemcpy(h_array, d_array, sizeof(void*) * N,
      cudaMemcpyDeviceToHost);
cudaFree(d_array); free(h_array);
```

Técnicas GPGPU

- Eliminar laços

```
A = rand(4);
output = zeros(4);
for n=1:4
    output(n,:) = input(n,:) / n;
end
```

```
A=rand(4);
output=zeros(4);
output = A ./
         repmat([1:4].',[1,4]);
```

- Manter-se na memória de GPU

```
??? Error using ==> gpuArray at 28
Out of memory on device. You requested: 762.94Mb, device has 1.31Gb free
>> reset(d);
```

- Instruções atômicas

Técnicas GPGPU

- Arrayfun

```
>> A = gpuArray(rand(100,4));
>> output = arrayfun(@myFunction,A(:,1),A(:,2),A(:,3),A(:,4));
```

- Criar um *kernel* (PTX)

```
function out=myFunction(a1,a2,a3)
out = (a1+a2+a3) / 3;
```

- Eliminar gargalos

Usar *matlabpool* para multi-GPU

```
myCluster = parcluster('local');
myCluster.NumWorkers = 4;
matlabpool(myCluster);
parfor i=1:4
    gpuDevice(i);

    % código para cada GPU

end
```

Usar *matlabpool* para multi-GPU

```
spmd;
     id=labindex;
     if (id < 3) G=gpuDevice(id);
          A=gpuArray.rand(1024,1024);
          if (id ==1) B=fft(del2(A));
          else B=fft(A);
          end
          B=abs(B);
     else disp('no GPUs')
     end
end
```

Usar *matlabpool* para multi-GPU

```
>>whos

Name  Size  Bytes  Class  Attributes
A     1x2   697    Composite
B     1x2   697    Composite
g     1x2   697    Composite
id    1x2   697    Composite
p     1x2   697    Composite

>>
mesh(cell2mat(B(1)));mesh(cell2mat(B(2)));
```

Matlab: Primos

```
>> clear

>> n=4E8;

>> x=rand(1,n);
>> y=rand(1,n);

>> tic; m = sum(sqrt(x.^2+y.^2)<=1); toc
Elapsed time is 2.064938 seconds.

>> pimc=4*m/n;
>> disp([pi; pimc])
   3.141592653589793
   3.141577000000000
```

Matlab: Primos

- Kernel

```
function n = primosKernel(n)
% MATLAB Kernel
% boolean primo = testprgpu(numero)
x=1+sqrt(n)/6;
for k=1:x
    if mod(n,6*k-1)==0||mod(n,6*k+1)==0,
        n=0;
        return;
    end
end
```

Matlab: Primos

- Principal

```
% prepare data
v=11:2:99999999;
disp(['Array size = ' (size(v,2)) ' of class ' class(v) ]);
tic; p=arrayfun(@primosKernel,v);   % Evaluate on CPU
cpuTime=toc,
primosCPU=v(p>0);

g=gpuArray(v);       % Copy data to the GPU
tic; q=arrayfun(@primosKernel,g);   % Evaluate on GPU
gpuTime=toc,
primosGPU=v(gather(q)>0);

disp([ num2str(size(r,2)) ' primes found;  Su = Ts/Tp = '...
    num2str(cpuTime,2) '/' num2str(gpuTime,2) ' = '...
    num2str(cpuTime/gpuTime,4)]),
```

Matlab: Primos

- Execução

```
>> primos
Array size = 4533330 of class double

cpuTime =  124.8238
gpuTime =    4.3501

1091309 primes found;  Su = Ts/Tp = 1.2e+02/4.4 = 28.69
```

Matlab: Primos

- Kernel

```
function n = primosKernel(n)
% MATLAB Kernel
% boolean primo = testprgpu(numero)
x=1+sqrt(n)/6;
for k=1:x
    if mod(n,6*k-1)==0||mod(n,6*k+1)==0,
        n=0;
        return;
    end
end
```

Matlab: Primos

- Kernel

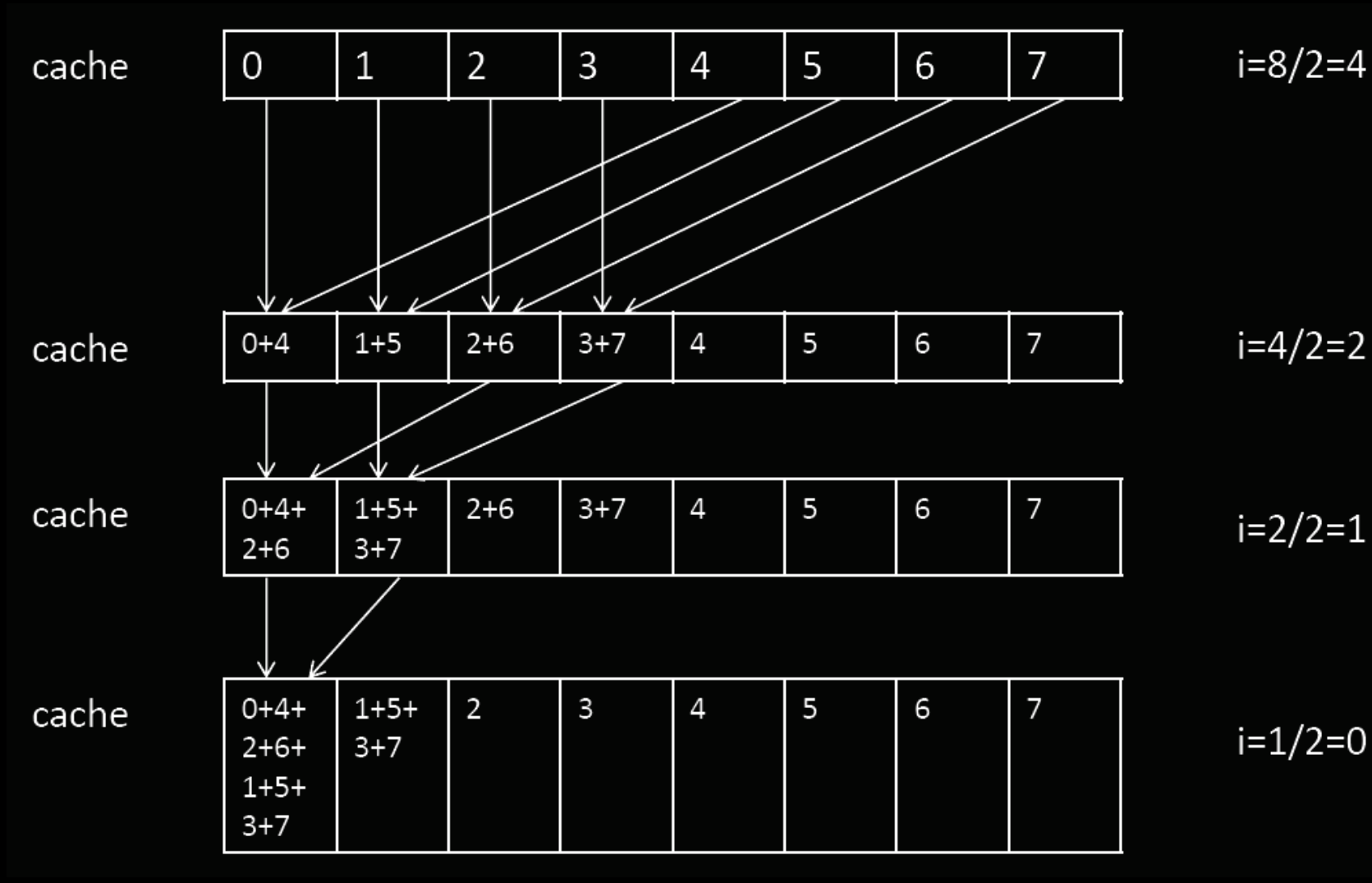
```
function primo = testprgpu(n)
% MATLAB Kernel
% boolean primo = testprgpu(numero)
primo=true;
for k=1:sqrt(n)/6
    if mod(n,6*k-1)==0,
        primo=false;
        break;
    elseif mod(n,6*k+1)==0,
        primo=false;
        break;
    end
end
```

```
disp('Testing double precision')
tic; p=arrayfun(@testprgpu,v);     % Evaluate on CPU
primosCPU=v(p);
cpuTime=toc
tic; g= gpuArray(v);               % Copy data to GPU
q=arrayfun(@testprgpu,g);          % Evaluate on GPU
primosGPU=v(gather(q));
gpuTime=toc
speedup=cpuTime/gpuTime
disp('Testing single precision')
v=single(v);
tic;
p=arrayfun(@testprgpu,v);          % Evaluate on CPU
primosCPU=v(p);
cpuTime=toc
tic;
g= gpuArray(v);                    % Copy data to GPU
q=arrayfun(@testprgpu,g);          % Evaluate on GPU
primosGPU=v(gather(q));
gpuTime=toc
speedup=cpuTime/gpuTime
```

Matlab: Primos benchmark

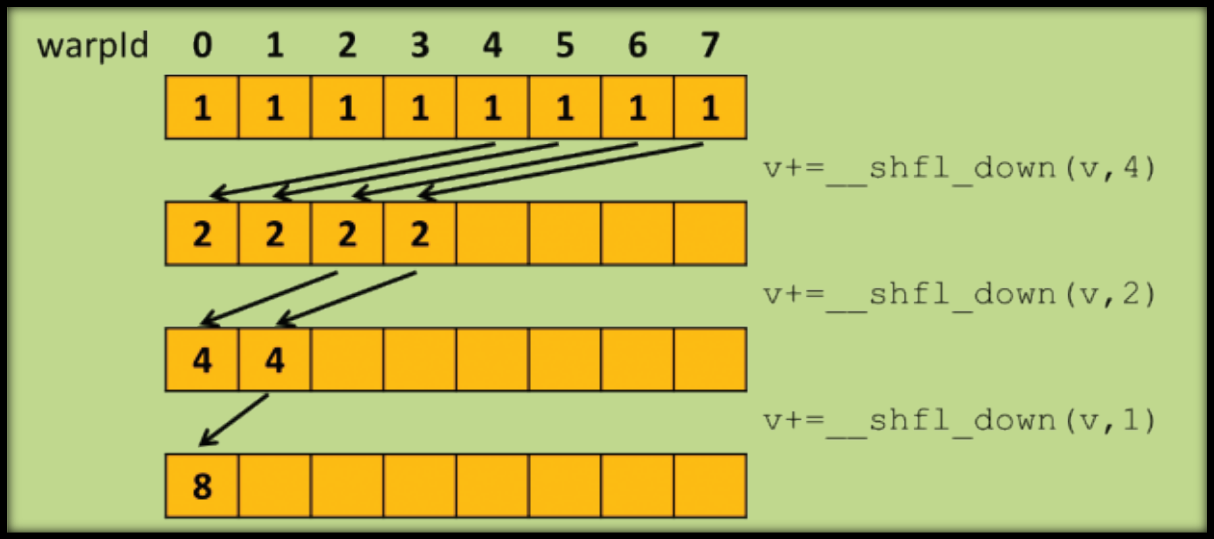
```
>> pribench
Testing double precision

cpuTime =  186.8746
gpuTime =    0.4944

speedup =  377.9515

Testing single precision

cpuTime =  200.4012
gpuTime =    0.5676

speedup =  353.0982
```

Redução (reduction)



Shuffle Warp Reduce

```
__inline__ __device__
int warpReduceSum(int val) {
  for (int offset = warpSize/2; offset > 0; offset /= 2)
    val += __shfl_down(val, offset);
  return val;
}
```



Recursos educacionais

- www.udacity.com/course/cs344
- developer.nvidia.com/cuda-education-training
- devblogs.nvidia.com/parallelforall/
- bit.ly/cudacasts

Referências

Programs, Documents and Labs

- www.mathworks.com/discovery/matlab-gpu.html
- developer.nvidia.com/cudazone
- nvidia.qwiklab.com
- docs.nvidia.com

Forums

- devtalk.nvidia.com
- stackoverflow.com/cuda