GPU: A matriz de processadores

Paulo Henrique Junqueira Amorim
Francisco Romulo da Silva Araújo

Instituto de Computação - Unicamp

25 de junho de 2011

Ano	Tipo	Modelo	Trans.	Perf.	Memory Band.
2004	GPU	GeForce 6800	222 Mio.	53 GFLOPS	35.2 GB/sec
	CPU	P4 570 (3.8GHz)	125 Mio.	15 GFLOPS	6.4 GB/sec
2006	GPU	GeForce 8800	681 Mio.	518 GFLOPS	86.4 GB/sec
	CPU	Core 2 Extreme	582 Mio.	85 GFLOPS	12.8 GB/sec
2008	GPU	GTX - 280	1400 Mio.	933 GFLOPS	113.3 GB/sec
	CPU	Core i7-965	731 Mio.	102 GFLOPS	25.6 GB/sec

fonte: http://forums.nvidia.com

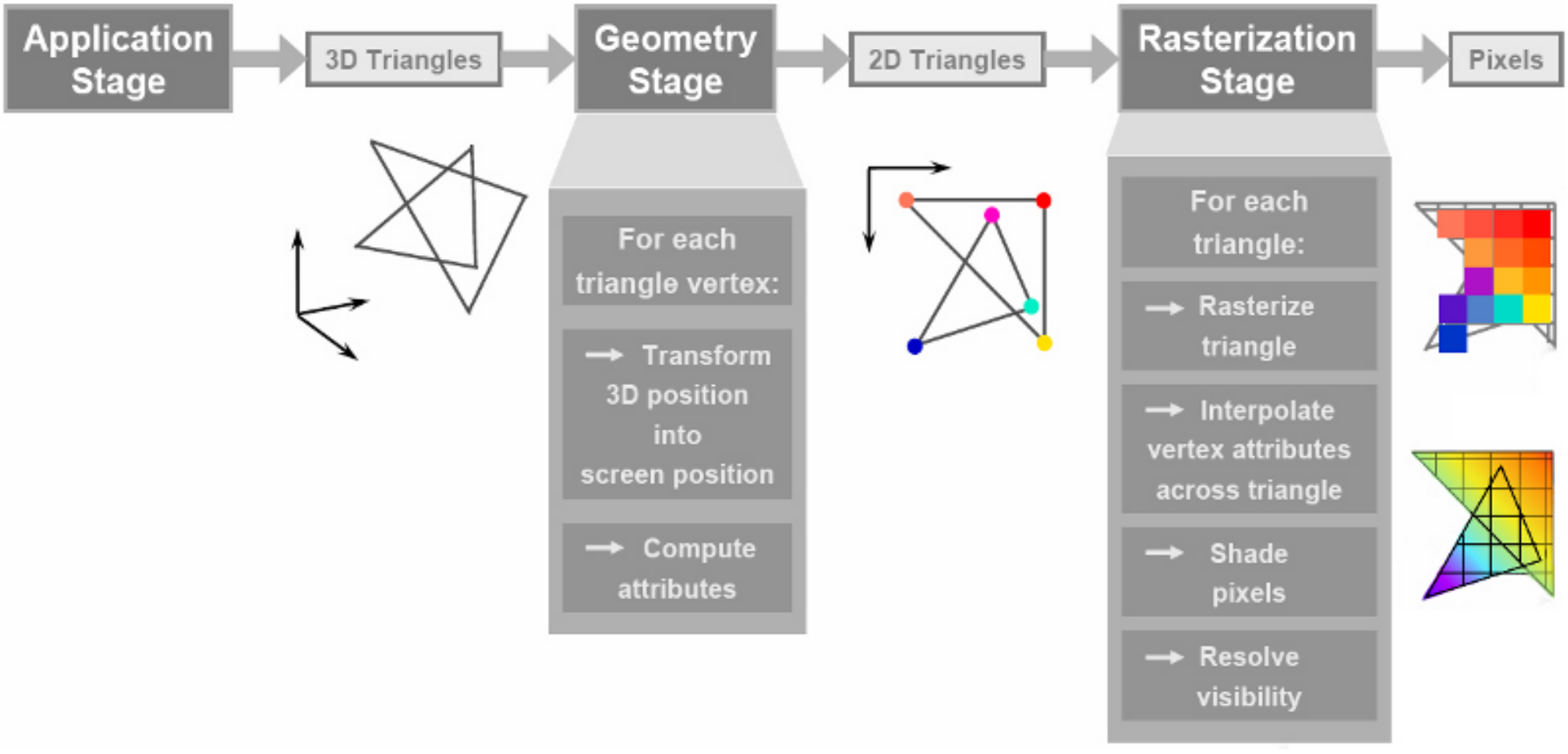

Imagem extraída e adaptada de [2]

CPU - Controlador VGA

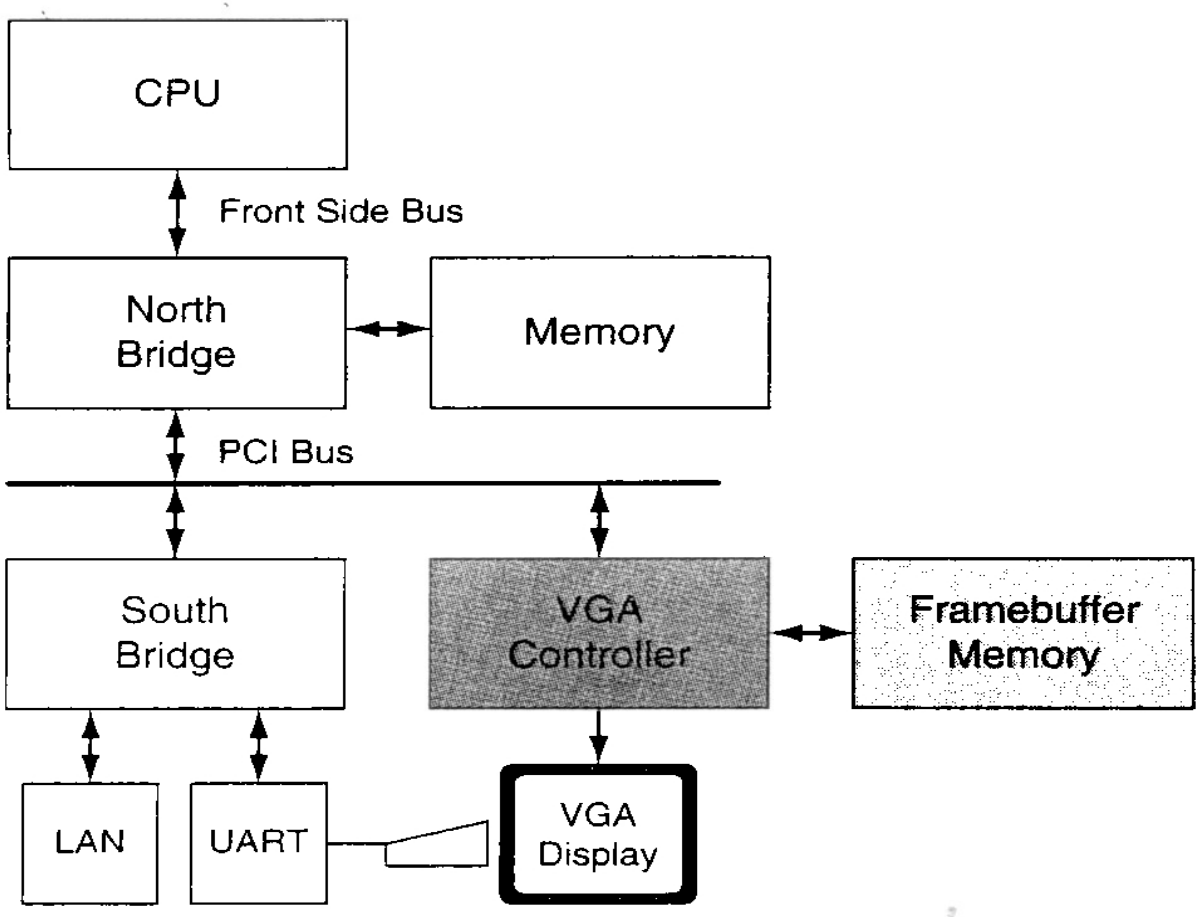

Imagem extraída de [3]

GPU & Arquitetura Intel

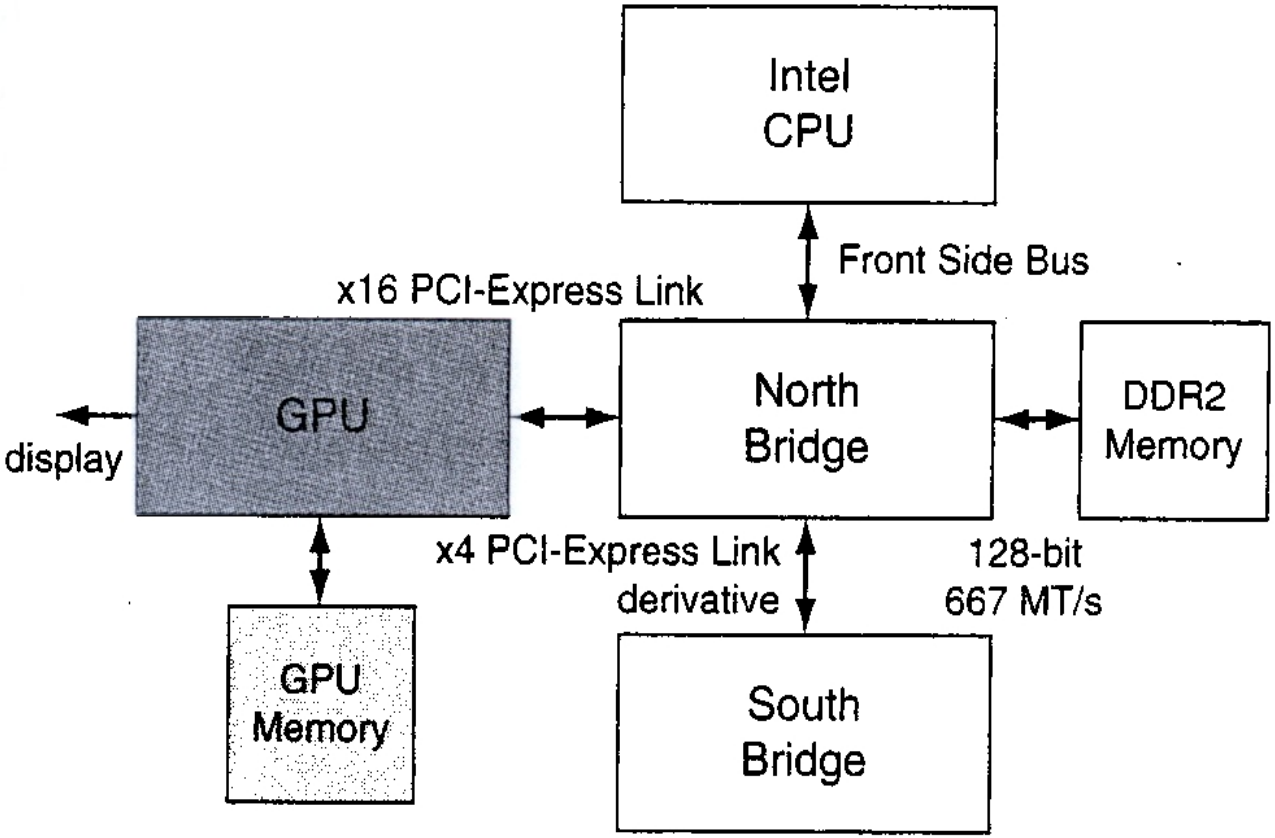

Imagem extraída de [3]

GPU & Arquitetura AMD

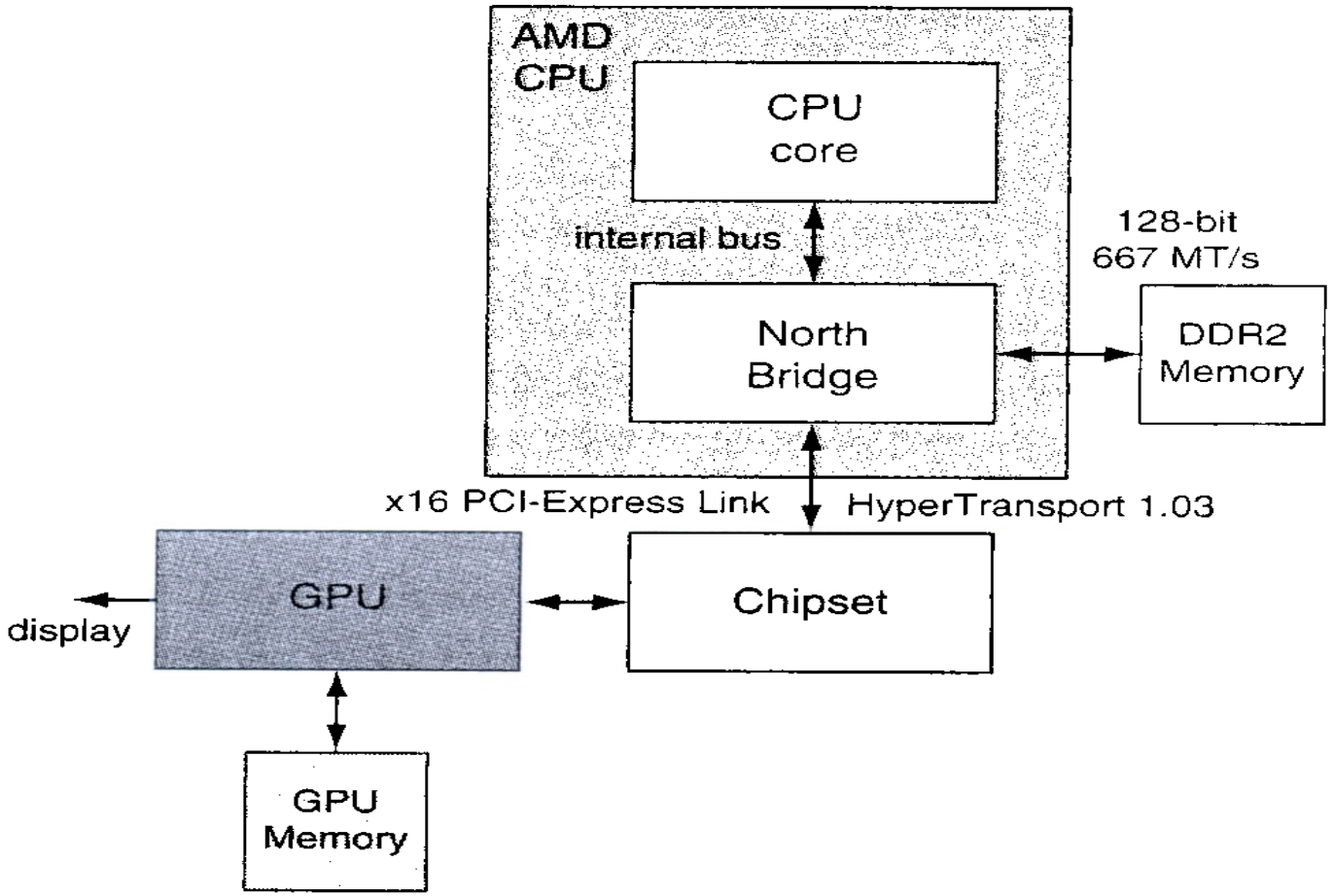

Imagem extraída de [3]

Streaming processor - SP

- Inteiro
- Ponto Flutuante
- Pode executar 4 *threads*

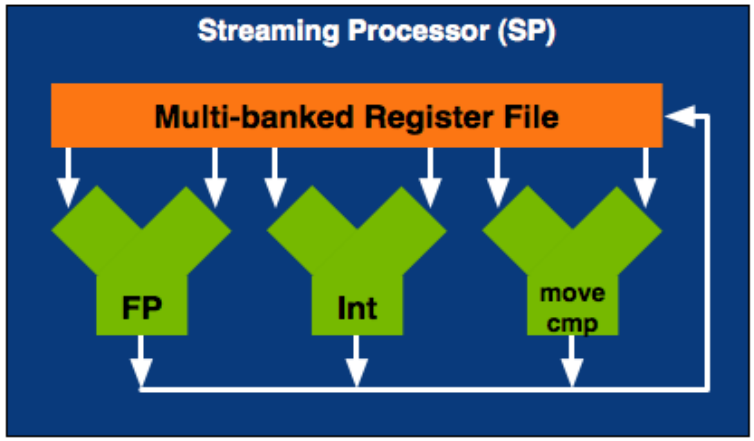

Imagem extraída de [1]

Streaming multiprocessors - SM

- 8 SP's
- 2 SFU ($\sin, \cos, \sqrt{9}, \log$)
- Cache Constante
- Cache Instrução
- 16 KB memória compartilhada

Imagem extraída de [1]

Texture processor cluster - TPC

- 3 SM's
- 8 KB texture cache

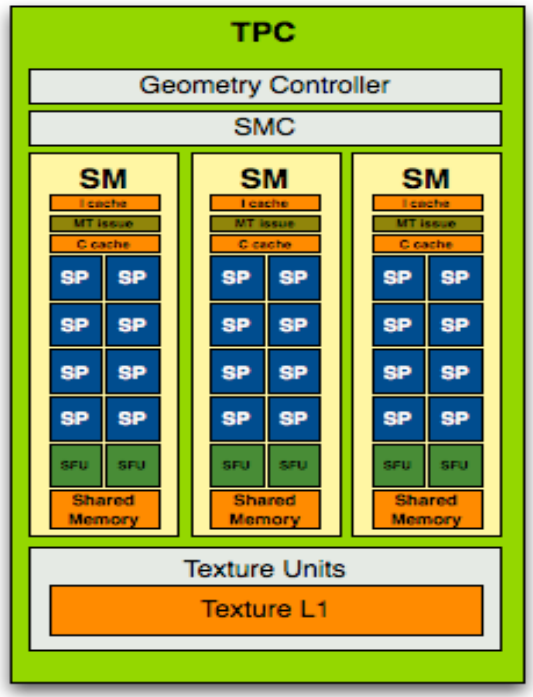

Imagem extraída de [1]

A Matriz de processadores

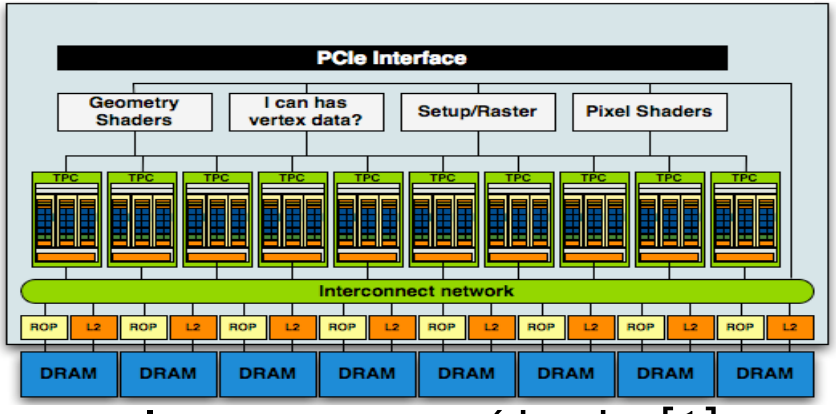

Imagem extraída de [1]

- 10 TPC's
- DRAM
- Cache L2
- ROP - (texture cache p/a DRAM, Interpolação)

*R*esumo

- 8 SP's - *Streaming processor*
- 3 SM's - *Streaming multiprocessors*
- 10 TPC's - *Texture processor cluster*
- $8x3x10 = 240$ processadores
- $8x3x10 = 240x4$ threads $= 960$ processos

Opções

- CUDA (NVidia)
- OpenCL (Consórcio empresas)

ANSI C x CUDA

```c
void add_matrix
(float* a, float* b, float* c, int N)
{
  int index;
  for(int i = 0; i < N; i++)
    for(int j = 0; j < N; j++)
    {
      index = i + j*N;
      c[index] = a[index] + b[index];
    }
}

int main()
{
   add_matrix( a, b, c, N);
}
```

```c
__global__ add_matrix
(float* a, float* b, float* c, int N)
{
 int i = blockIdx.x * blockDim.x + threadIdx.x;
 int j = blockIdx.y * blockDim.y + threadIdx.y;
 int index = i + j*N;
 if (i < N && j < N)
   c[index] = a[index] + b[index];
}

int main()
{
 dim3 dimBlock(blocksize, blocksize);
 dim3 dimGrid(N/dimBlock.x, N/dimBlock.y);
 add_matrix<<<dimGrid, dimBlock>>>(a,b,c,N);
}
```

Intel Larrabbe

- x86
- Cache L3 texture
- 16 Unidades Processamento Vetorial

Intel Larrabbe

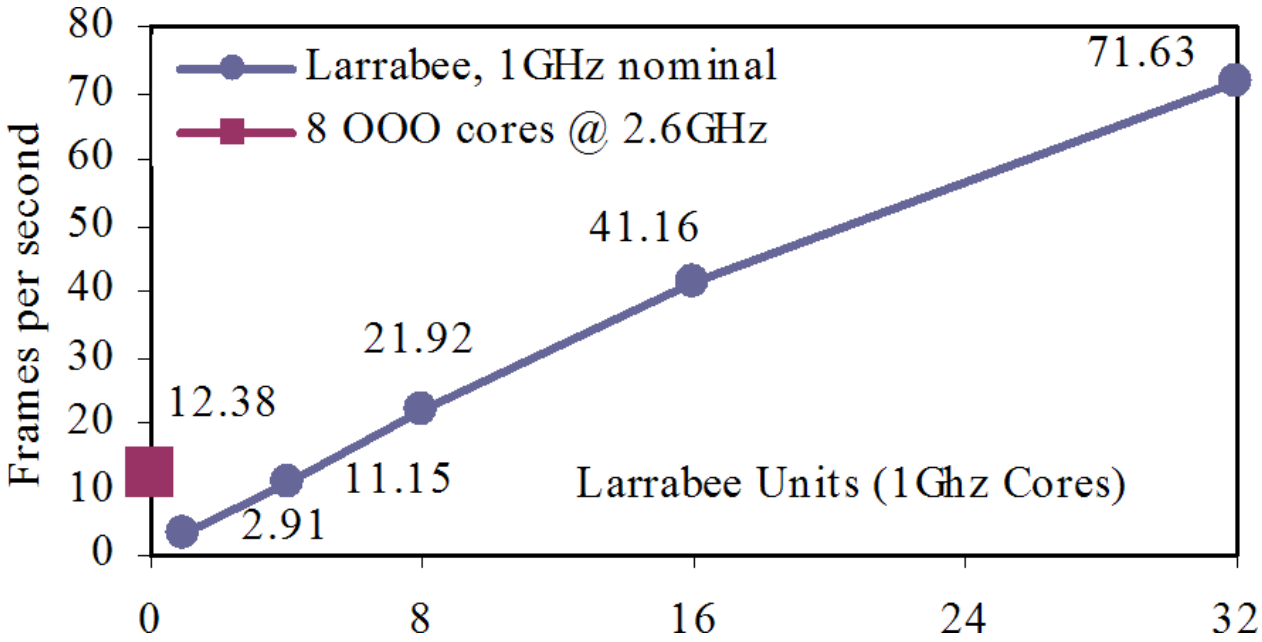

Imagem extraída de [4]

- Filtragem Domínio da Freq.

FFT

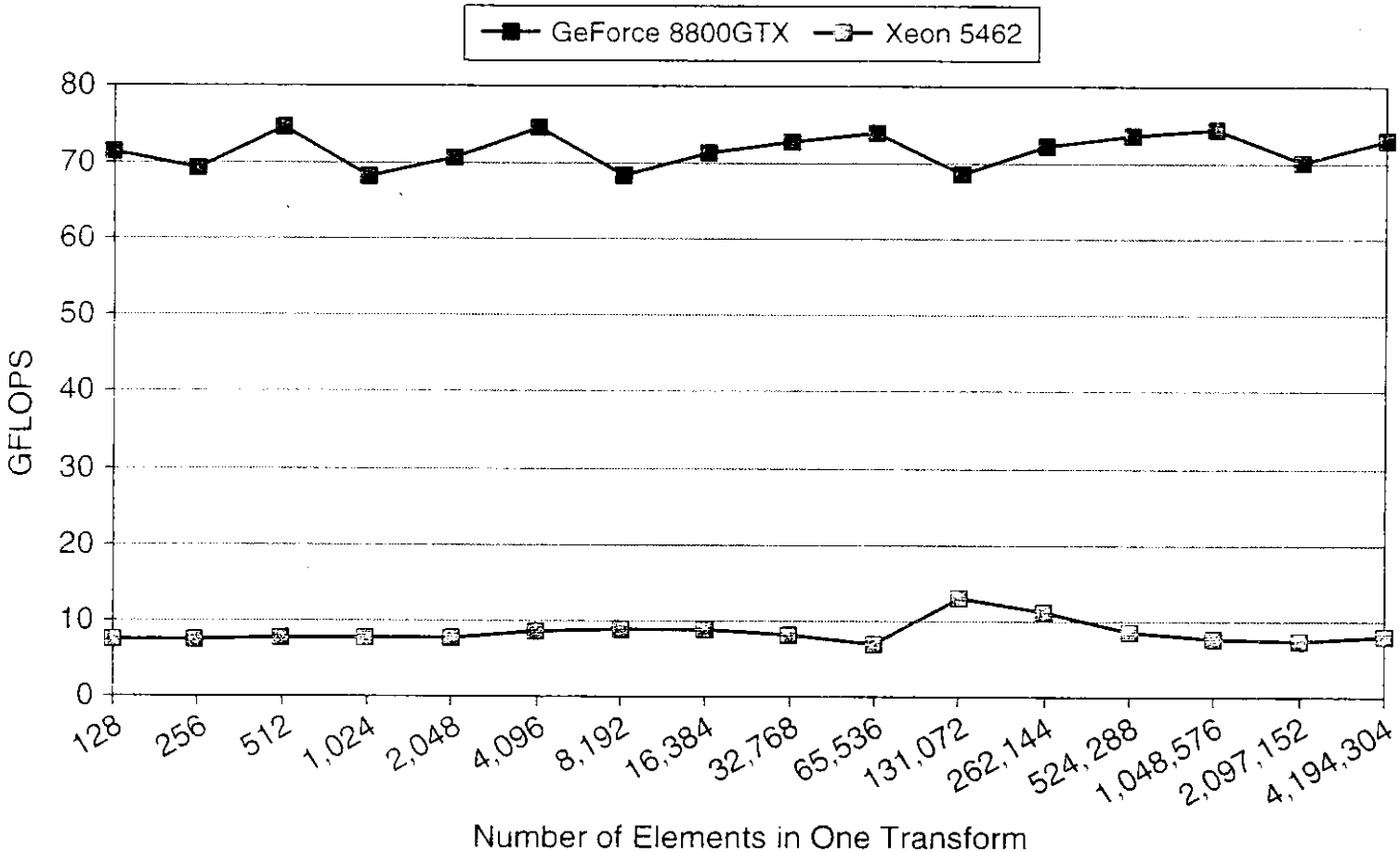

Imagem extraída de [3]

- Filtragem Domínio da Freq.
- Aplicações Paralelizáveis

Radix sort

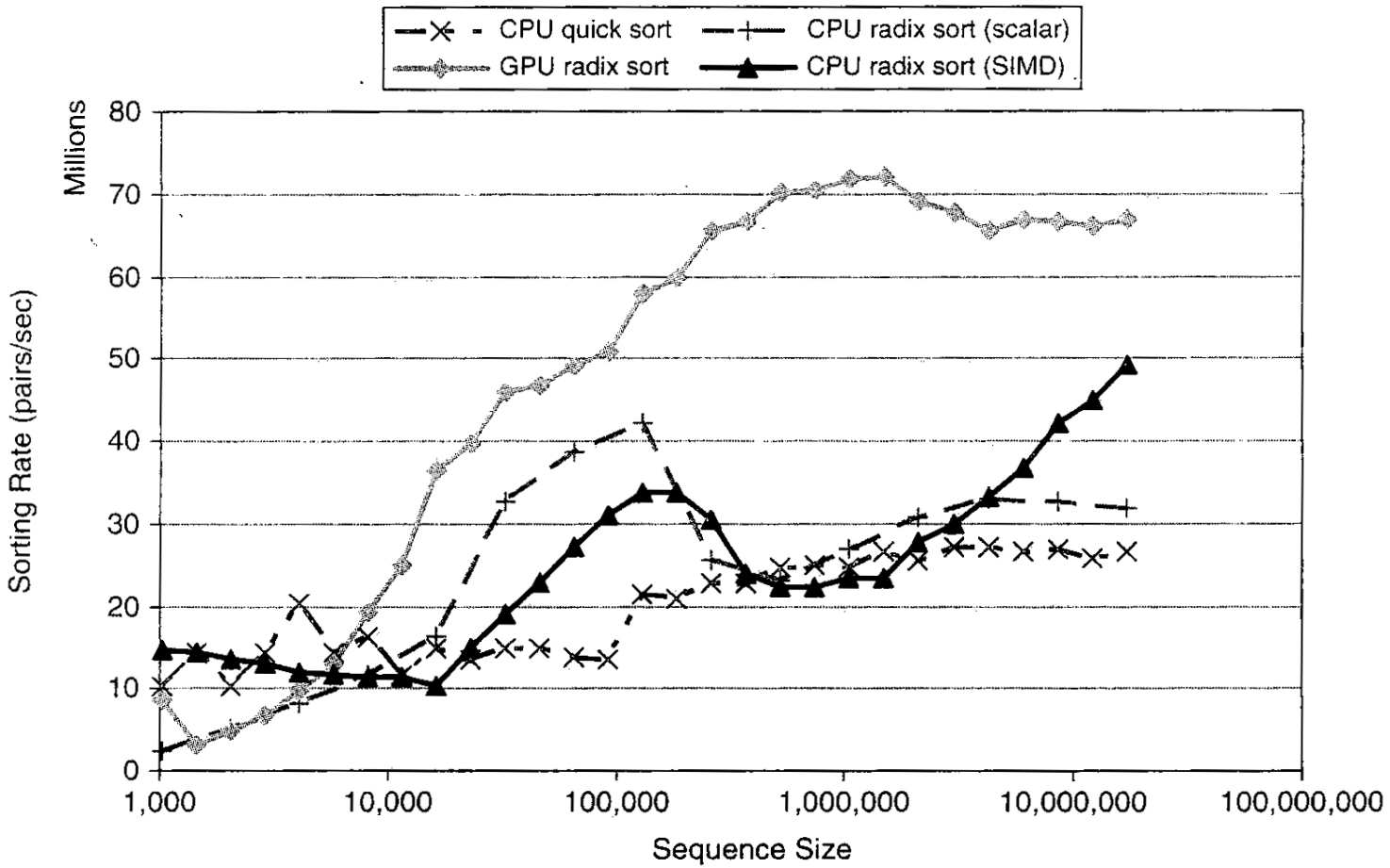

Imagem extraída de [3]

References I


AnandTech - NVIDIA's 1.4 billion transistor GPU: GT200 arrives as the GeForce GTX 280 & 260.
http://www.anandtech.com/show/2549/2, 6 2011.


Minh Tri Do Dinh.
Gpus - graphics processing units.
Vertiefungsseminar Architektur von Prozessoren, SS 2008, pages 3 – 4, 2008.


D.A. Patterson and J.L. Hennessy.
Computer organization and design: the hardware/software interface.
The Morgan Kaufmann Series in Computer Architecture and Design. Elsevier Morgan Kaufmann, 2008.

References II


Larry Seiler, Doug Carmean, Eric Sprangle, Tom Forsyth, Michael Abrash, Pradeep Dubey, Stephen Junkins, Adam Lake, Jeremy Sugerman, Robert Cavin, Roger Espasa, Ed Grochowski, Toni Juan, and Pat Hanrahan.
Larrabee: a many-core x86 architecture for visual computing.
ACM Trans. Graph., 27:18:1–18:15, August 2008.

Perguntas ?