

GPU Computing

Numerical Simulation - from Models to Software

Andreas Barthels

JASS 2009, Course 2, St. Petersburg, Russia

Prof. Dr. Sergey Y. Slavyanov
St. Petersburg State University

Prof. Dr. Thomas Huckle
Technische Universität München

March 29 – April 7, 2009



Outline

- 1 Introduction
- 2 GPU Hardware
 - GPU Functionality
 - GPU–CPU Comparison
- 3 GPU Computing Today
 - BrookGPU
 - OEM SDKs
 - Applications
- 4 GPU Computing Example
 - 2D Simplified Fluid Simulation
- 5 Outlook
 - DirectX 11
 - OpenCL
 - Heterogeneous Computing
- 6 Summary



Introduction

GPU Hardware

- highly parallel
- 1 Tflops for $< 120 \text{ €}$ ($\hat{=}$ 5,500 RUB)
- increasingly easy to program

This talk:

Applicability to numerical simulation and immediate visualization?

Outline

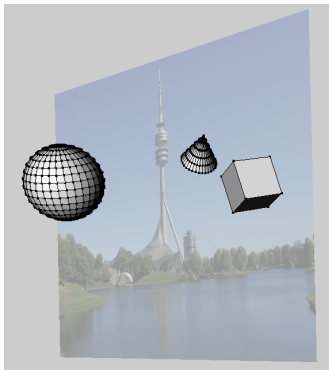
- 1 Introduction
- 2 GPU Hardware
 - GPU Functionality
 - GPU–CPU Comparison
- 3 GPU Computing Today
 - BrookGPU
 - OEM SDKs
 - Applications
- 4 GPU Computing Example
 - 2D Simplified Fluid Simulation
- 5 Outlook
 - DirectX 11
 - OpenCL
 - Heterogeneous Computing
- 6 Summary



GPU Functionality I

Standard Tasks

- display of graphical models and scenes on a computer
- works with polygons, vertices, textures



GPU Functionality II

Standard Task: Shading of Pixels

- The same code fragment for every pixel (\Rightarrow *SPMD*)
- Typically a pixel's value is independent on the others'
- \Rightarrow massive gain through parallelization

Recent Extensions

- increased programmability for more realistic visual effects
- simulation of rigid, deformable or fluid objects in game physics
- \Rightarrow general purpose programmability

GPU Functionality III

Programming Interfaces & Terminology

DirectX

- Vertex Shader
- Pixel Shader
- Geometry Shader

OpenGL

- Vertex Shader
- Fragment Shader
- (Geometry Shader)

GPU Functionality IV



Demo of AMD RenderMonkey

Outline

- 1 Introduction
- 2 GPU Hardware**
 - GPU Functionality
 - **GPU–CPU Comparison**
- 3 GPU Computing Today
 - BrookGPU
 - OEM SDKs
 - Applications
- 4 GPU Computing Example
 - 2D Simplified Fluid Simulation
- 5 Outlook
 - DirectX 11
 - OpenCL
 - Heterogeneous Computing
- 6 Summary



GPU-CPU Comparison

x86 CPU

- Multi Purpose
- OS Instructions
- One ALU per Core
- Mild SIMD per Core
- Slow Memory
- Big Cache

GPU

- Specialized to Graphics
- No OS Concepts
- +100 Parallel ALUs
- Extensive MIMD
- Fast Memory
- Small Cache

Peak Performance

AMD/ATI Radeon HD 4870 – ≈ 180 €

- 1,200 MADD Gflops (single precision) – 240 Gflops
- 115.2 GBytes/s Memory Bandwidth

NVIDIA Geforce GTX 285 – ≈ 330 €

- 1,063 MADD Gflops (single precision) – 78 Gflops
- 159.0 GBytes/s Memory Bandwidth

Intel Core i7 965 XE – ≈ 1000 €

- ≈ 70 M+ADD Gflops (single precision) – ≈ 35 Gflops
- ≈ 25.6 GBytes/s Memory Bandwidth

(\implies What about double precision?)



Peak Performance

AMD/ATI Radeon HD 4870 – ≈ 180 €

- 1,200 MADD Gflops (single precision) – 240 Gflops
- 115.2 GBytes/s Memory Bandwidth

NVIDIA Geforce GTX 285 – ≈ 330 €

- 1,063 MADD Gflops (single precision) – 78 Gflops
- 159.0 GBytes/s Memory Bandwidth

Intel Core i7 965 XE – ≈ 1000 €

- ≈ 70 M+ADD Gflops (single precision) – ≈ 35 Gflops
- ≈ 25.6 GBytes/s Memory Bandwidth

(\implies What about double precision?)



Peak Performance

AMD/ATI Radeon HD 4870 – ≈ 180 €

- 1,200 MADD Gflops (single precision) – **240 Gflops**
- 115.2 GBytes/s Memory Bandwidth

NVIDIA Geforce GTX 285 – ≈ 330 €

- 1,063 MADD Gflops (single precision) – **78 Gflops**
- 159.0 GBytes/s Memory Bandwidth

Intel Core i7 965 XE – ≈ 1000 €

- ≈ 70 M+ADD Gflops (single precision) – **≈ 35 Gflops**
- ≈ 25.6 GBytes/s Memory Bandwidth

(\implies What about **double precision**?)



Outline

- 1 Introduction
- 2 GPU Hardware
 - GPU Functionality
 - GPU–CPU Comparison
- 3 GPU Computing Today**
 - **BrookGPU**
 - OEM SDKs
 - Applications
- 4 GPU Computing Example
 - 2D Simplified Fluid Simulation
- 5 Outlook
 - DirectX 11
 - OpenCL
 - Heterogeneous Computing
- 6 Summary



BrookGPU I

- BrookGPU was the first GPGPU framework
- Developed at Stanford
- Extension to C
- Vector datatypes:
 - `float, float2, float3, float4`
 - `int, int2, int3, int4`
 - ...
 - `double, double2`



BrookGPU II

- Streams

`float s<10, 10>;` – 2 dimensional 10x10 float matrix
Access to stream items is restricted and regularized

- Kernels

```
kernel void k(float s<>, float3 f,  
              float a[10][10], out float o<>) {...
```



BrookGPU III

- Reduction Operations on Streams

```
void reduce sum (float a<>,
                reduce float result<>)
    result = result + a;
```

- Scatter & Gather Ops on Streams

```
streamScatterOp(s, ...)
streamGatherOp(s, ...)
```



Outline

- 1 Introduction
- 2 GPU Hardware
 - GPU Functionality
 - GPU–CPU Comparison
- 3 GPU Computing Today**
 - BrookGPU
 - OEM SDKs**
 - Applications
- 4 GPU Computing Example
 - 2D Simplified Fluid Simulation
- 5 Outlook
 - DirectX 11
 - OpenCL
 - Heterogeneous Computing
- 6 Summary



AMD/ATI: Stream SDK I

Custom Operations

AMD relies on Brook+, an extension to BrookGPU

AMD Core Math Library

- Optimized implementation of BLAS
- Can use multiple CPU cores and multiple GPUs



AMD/ATI: Stream SDK II

Stream KernelAnalyzer - Brook+

File Edit Help

Source Code
Function **k**

```

1 // Enter your kernel in this window
2 kernel void k(float s, float x<>, float y<>, out float z<>)
3 {
4   z = s*x+y;
5 }
6

```

Object Code
Format: Radeon HD 4870 (RV770) Assembly

```

; ----- Disassembly -----
00 TEX: ADDR(48) CNT(2) VALID_PIX
   0 SAMPLE R1.x____, R0.yxxx, t0, #0 UNNORM(XYZW)
   1 SAMPLE R0.x____, R0.yxxx, t1, #0 UNNORM(XYZW)
01 ALU: ADDR(32) CNT(3) KCACHE0(CB0:0-15)
   2 y: MOV          R1.y, 0.0f
   z: MUL_w        _____, KCO[0].x, R1.x
   3 x: ADD          R1.x, R0.x, PVZ.z
02 EXP_DONE: PIX0, R1.xyyy
END_OF_PROGRAM

```

Compiler Statistics (Using CAL 9.1)

Name	GPR	Scratch Reg	Min	Max	Avg	Est Cycles	ALU:Fetch	BottleNeck	Thread/Clock	Throughput
Radeon HD 2900	3	0	2.00	2.80	2.27	2.00	0.50	Texture Fetch	8.00	5936 M Threads/Sec
Radeon HD 2400	3	0	2.00	2.80	2.27	2.00	1.00	ALU Ops	2.00	1600 M Threads/Sec
Radeon HD 2600	3	0	2.00	2.80	2.27	2.00	0.50	Texture Fetch	2.00	1600 M Threads/Sec
Radeon HD 3870	3	0	2.00	2.80	2.27	2.00	0.50	Texture Fetch	8.00	6200 M Threads/Sec
Radeon HD 4870	3	0	1.00	1.12	1.00	1.00	1.25	Global Write	16.00	12000 M Threads/Sec
Radeon HD 4670	3	0	1.00	1.40	1.13	1.00	1.00	Global Write	8.00	6000 M Threads/Sec
FireStream 9170	3	0	2.00	2.80	2.27	2.00	0.50	Texture Fetch	8.00	6200 M Threads/Sec
FireStream 9250	3	0	1.00	1.12	1.00	1.00	1.25	Global Write	16.00	10000 M Threads/Sec
FireStream 9270	3	0	1.00	1.12	1.00	1.00	1.25	Global Write	16.00	12000 M Threads/Sec

Compiler Output

Demo of Brook+ and AMD Stream KernelAnalyzer



NVIDIA: CUDA SDK

Custom Operations

CUDA is very similar to Brook (streams, kernels, . . .)

Math Libraries

- CUBLAS: BLAS implemented in CUDA
- CUFFT: Fourier Transforms on GPU

Outline

- 1 Introduction
- 2 GPU Hardware
 - GPU Functionality
 - GPU–CPU Comparison
- 3 GPU Computing Today**
 - BrookGPU
 - OEM SDKs
 - Applications**
- 4 GPU Computing Example
 - 2D Simplified Fluid Simulation
- 5 Outlook
 - DirectX 11
 - OpenCL
 - Heterogeneous Computing
- 6 Summary



Applications using GPU Computing

- Image processing
- Video de-/encoding
- Protein folding (folding@home)
- SETI@home

Outline

- 1 Introduction
- 2 GPU Hardware
 - GPU Functionality
 - GPU–CPU Comparison
- 3 GPU Computing Today
 - BrookGPU
 - OEM SDKs
 - Applications
- 4 GPU Computing Example**
 - 2D Simplified Fluid Simulation**
- 5 Outlook
 - DirectX 11
 - OpenCL
 - Heterogeneous Computing
- 6 Summary



2D Simplified Fluid Simulation I

Denote u to be the height of the water surface at a certain point in time and space and let c be the wave speed:

$$\frac{\partial^2 u}{\partial t^2} = c^2 \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right)$$

2D Simplified Fluid Simulation II

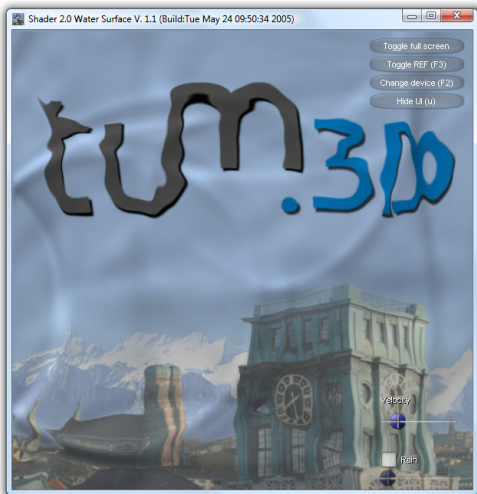
Discretizing space and time each equidistantly by Δt and h :

$$\frac{u_{i,j}^{t+1} - 2u_{i,j}^t + u_{i,j}^{t-1}}{\Delta t^2} = c^2 \left(\frac{u_{i+1,j}^t + u_{i-1,j}^t + u_{i,j+1}^t + u_{i,j-1}^t - 4u_{i,j}^t}{h^2} \right)$$

Applying $u_{i,j}^t := 0.5 \cdot (u_{i,j}^{t+1} + u_{i,j}^t)$ yields system of linear equations.

Interactive Demo

2D simplified fluid simulation using conjugate gradient solver



Outline

- 1 Introduction
- 2 GPU Hardware
 - GPU Functionality
 - GPU–CPU Comparison
- 3 GPU Computing Today
 - BrookGPU
 - OEM SDKs
 - Applications
- 4 GPU Computing Example
 - 2D Simplified Fluid Simulation
- 5 Outlook**
 - **DirectX 11**
 - OpenCL
 - Heterogeneous Computing
- 6 Summary



DirectX 11

- DirectX drives the industry standard for GPUs
- DX 11 compliant hardware is designed to support GPGPU
- Object Oriented Programming in “Compute Shader”
- Great increase in flexibility
- Increased double precision performance
- Wait until Windows 7



Outline

- 1 Introduction
- 2 GPU Hardware
 - GPU Functionality
 - GPU–CPU Comparison
- 3 GPU Computing Today
 - BrookGPU
 - OEM SDKs
 - Applications
- 4 GPU Computing Example
 - 2D Simplified Fluid Simulation
- 5 **Outlook**
 - DirectX 11
 - **OpenCL**
 - Heterogeneous Computing
- 6 Summary



OpenCL

- Open standard, driven by Khronos group



- Not Object Oriented, just C, like Brook et al.
- Interoperability with OpenGL
- Abstraction layer for any type of hardware

Outline

- 1 Introduction
- 2 GPU Hardware
 - GPU Functionality
 - GPU–CPU Comparison
- 3 GPU Computing Today
 - BrookGPU
 - OEM SDKs
 - Applications
- 4 GPU Computing Example
 - 2D Simplified Fluid Simulation
- 5 Outlook**
 - DirectX 11
 - OpenCL
 - Heterogeneous Computing**
- 6 Summary



Heterogeneous Computing

- Software support via OpenCL
- GPU and CPU combined hardware soon to come
(⇒ Intel Larrabee, AMD Fusion)



Outline

- 1 Introduction
- 2 GPU Hardware
 - GPU Functionality
 - GPU–CPU Comparison
- 3 GPU Computing Today
 - BrookGPU
 - OEM SDKs
 - Applications
- 4 GPU Computing Example
 - 2D Simplified Fluid Simulation
- 5 Outlook
 - DirectX 11
 - OpenCL
 - Heterogeneous Computing
- 6 Summary

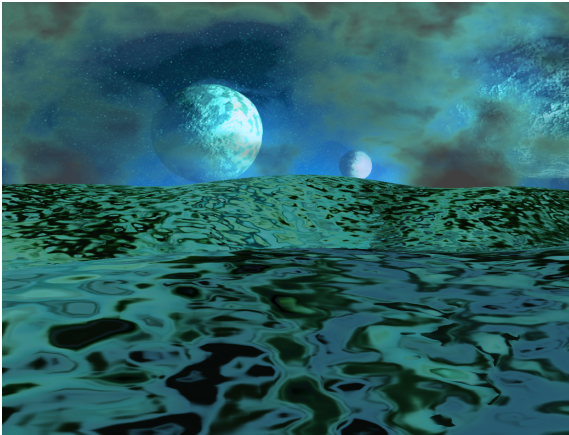


Summary

- GPUs feature high performance for little money
- Capable of descent immediate visualization
- Upcoming standards in hardware & APIs are promising



Thank you



NVIDIA and AMD are registered trademarks of NVIDIA and AMD Corp. respectively.

AMD RenderMonkey, the corresponding samples and the AMD Stream SDK are copyright AMD.

