Scientific Computing on a GPU Using CUDA

Georgia Institute of Technology
Center for Signal and Image Processing
Steve Conover
February 2009

Introduction

CUDA is a **tool** to turn your graphics card into a small **computing cluster**.

It's **not** always **easy** to use, but the **boost** it may provide to your processing power could make it **worthwhile**.

This is a **practical**, **application** oriented **software talk** to present some **ideas** on how to gain some **cheap**, **accessible** processing **power** when you need it.

CSIP

Overview

- What is **CUDA**?
- What can I **use** it for?
- Something **practical**

What is CUDA?

What is a GPU?

What is CUDA?

What can it do?

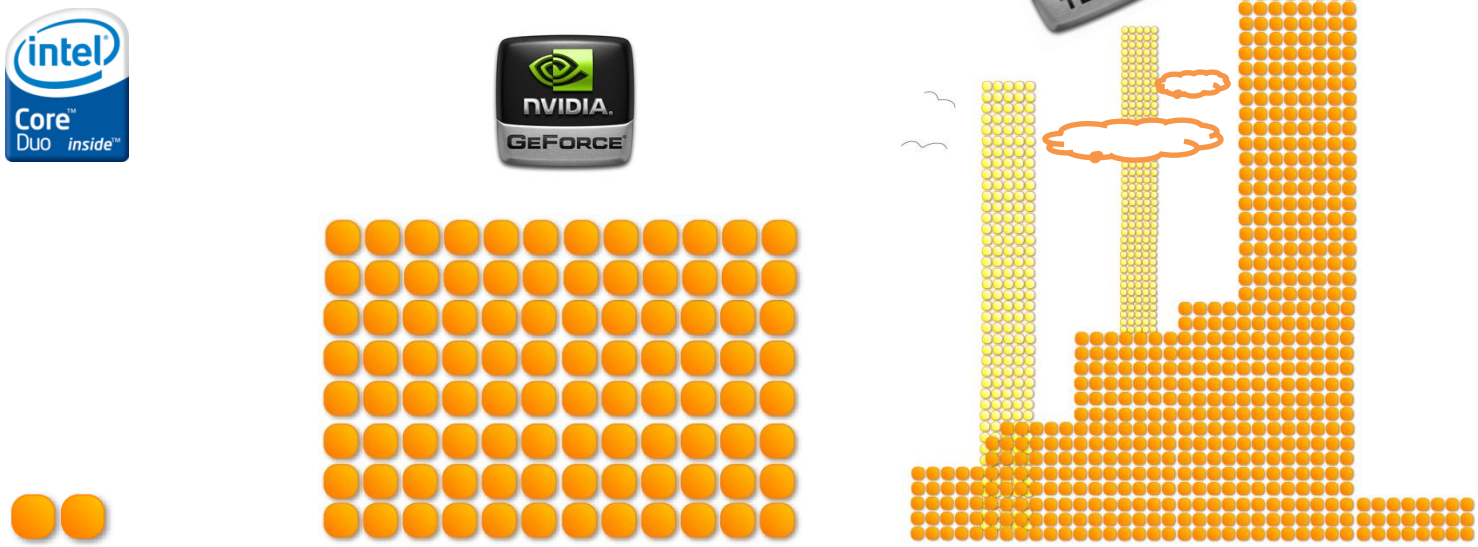
What is a GPU?

- **G**raphics **P**rocessing **U**nit (GPU)
- **Fast** and **specialized** for graphics
- Good with **pictures**. Good with **polygons**.
- **Parallel**. **Pipelined**.

MythBusters: What is a GPU?

General Computing with Parallelization++

- GPU **evolution** has brought us to the place where these **parallel** pipelines can execute **arbitrary instructions**.
- Why is this especially **interesting**?



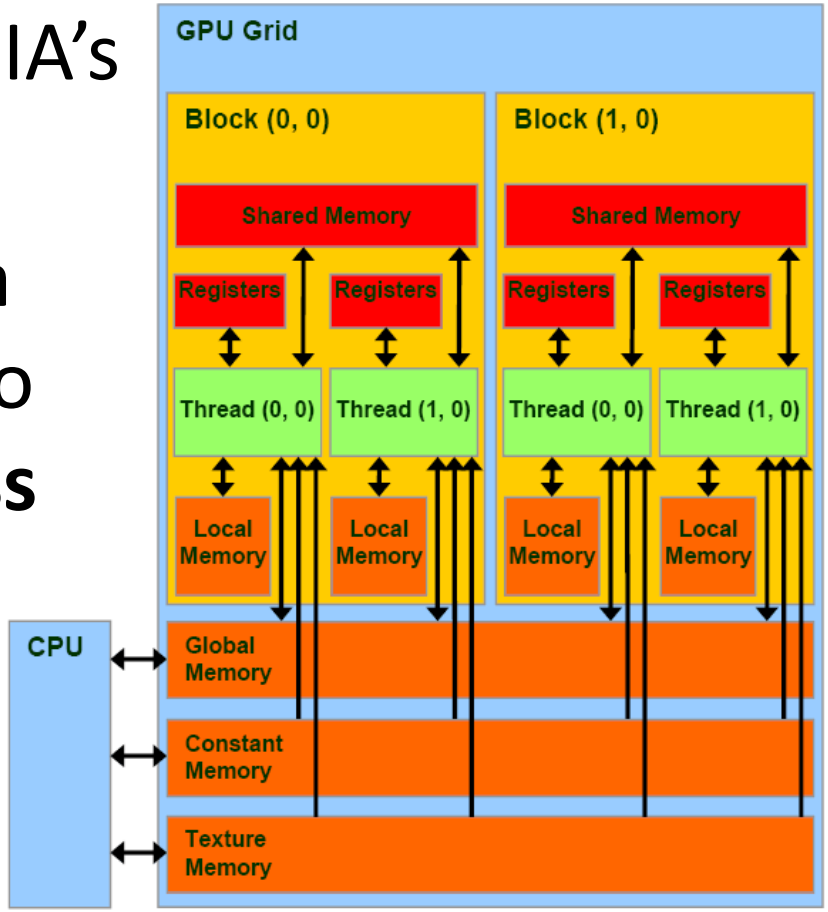
2 Cores

192 Cores (GTX260)

960 Cores (Tesla S1070)

What is CUDA?

- **CUDA** is a **c/c++ interface** for NVIDIA's graphics cards
- A way to give **data** and **instructions** to the card to **process in parallel**



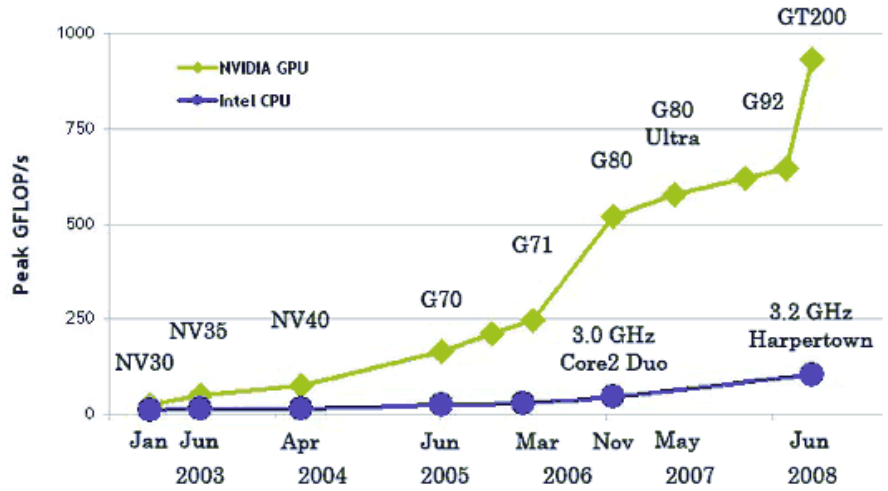
Example

```
__global__ void vecAdd(float* A,
    float* B, float* C)
{
    int i = threadIdx.x;
    C[i] = A[i] + B[i];
}

int main()
{
// Kernel invocation
vecAdd<<<1, N>>>(A, B, C);
}
```

Benefits of CUDA

- **Parallel = fast**
- Tightly **coupled** threads and memory
- Programmable with c-like **syntax**
- **Independent** of CPU
- Low hardware **cost**



The Bad News

- Takes **time** to learn
- Takes **time** to code
- Takes **time** to code fast
- Multithreaded applications can be **hard**
- Is often **not worth it**

What Can We Use it For?

CUDA is good for some things

Bad for others.

CUFFT

CUBLAS

Conjugate Gradient

Good Stuff

- **Linear Algebra**
- **FFT**
- **Independent** operations (parallelizable)
- **BIG** problems
- Time **intensive** problems

Bad Stuff

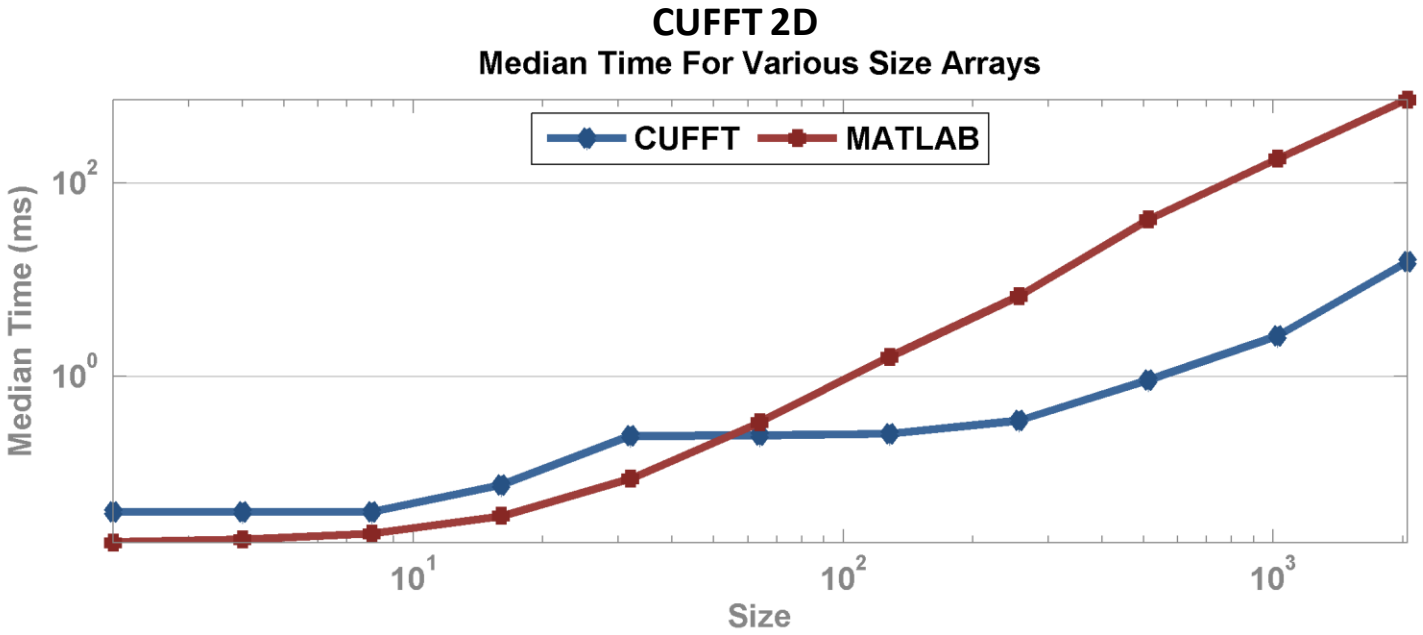
- Small Problems
- Quick Problems
- Problems that take a long time but we don’t care

Toys that Come with CUDA

- **CUBLAS**: A linear algebra package for executing matrix/vector operations.
- **CUFFT**: A library for calculating **1D**, **2D**, and **3D** Fast Fourier Transforms (**FFT**)

CUFFT

1. Make a CUFFT “Plan”
2. Move data to the card
3. Execute the Plan on the data



CUBLAS

CUDA **B**asic **L**inear **A**lgebra **S**ubprograms (CUBLAS)

1. Level 1: Vector Operations (norms, inner product, max, min, add, subtract)
2. Level 2: Matrix-Vector Operations (matrix-vector, multiply, outer product)
3. Level 3: matrix-matrix operations

With various support for dense, symmetric, and banded matrix forms.

Conjugate Gradient

An iterative method to solve the equation

$$Ax = b$$

Where A is symmetric and positive definite.

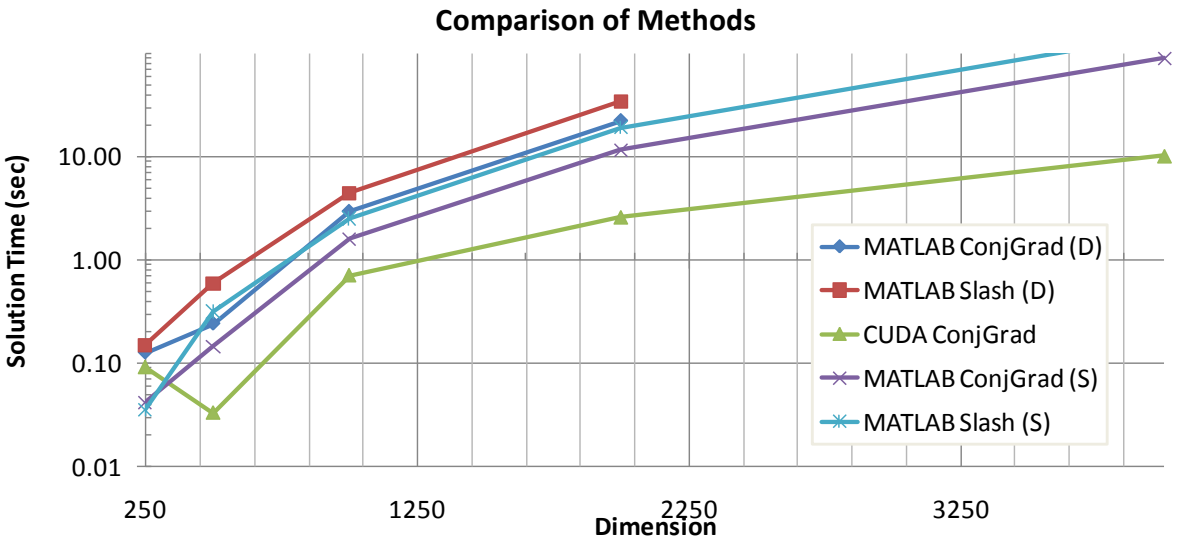
```
function [x ,count] = conjgrad(A,b,x0)
% Conjugate Gradient from Wikipedia:
% http://en.wikipedia.org/wiki/Conjugate_gradient

count = 0;
r = b - A*x0;
w = -r;
z = A*w;
a = (r'*w)/(w'*z);
x = x0 + a*w;
B = 0;

for i = 1:size(A,1);
    r = r - a*z;
    if( norm(r) < 1e-10 )
        break;
    end
    B = (r'*z)/(w'*z);
    w = -r + B*w;
    z = A*w;
    a = (r'*w)/(w'*z);
    x = x + a*w;
    count = count + 1;
end
```

Conjugate Gradient

Precision	Method	1000 250	2000 500	4000 1000	8000 2000	16000 4000
Double	MATLAB ConjGrad (D)	0.126	0.240	2.931	22.221	Memory
	MATLAB Slash (D)	0.150	0.589	4.448	34.266	Memory
Single	MATLAB ConjGrad (S)	0.0420	0.145	1.604	11.578	89.143
	MATLAB Slash (S)	0.036	0.319	2.506	18.87	150.0
Single	CUDA ConjGrad	0.091	0.0328	0.7110	2.599	10.20

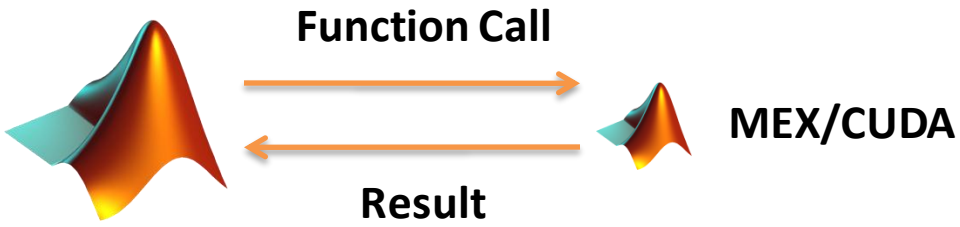


Other Projects on Cuda Zone

Something Practical

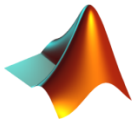
What is a Good Plan to Use CUDA?

- Simple MATLAB call, maybe:



- Full MATLAB to CUDA Conversion:

The Road to Conversion



MATLAB

1. Easy to prototype 2. High efficiency 3. Easy failures 4. Low barrier to entry 5. User community	1. Slow 2. Software costs money 3. Single threaded *

boost
C++ LIBRARIES

1. C++ is faster 2. Free 3. Boost has easy classes for BLAS 4. User community 5. Works on the CPU 6. Compares directly with GPU	1. Much lower coder efficiency

CUDA + Boost

1. Very fast 2. Offload processing to GPU	1. Parallel computing is hard 2. High barrier to entry 3. Lead time

CSIP

A Few Tricks

- **Encapsulate** CUDA in nice C++

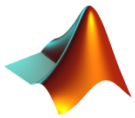
X
```
// fu1 = x - u
cublasScopy(N, d_x, 1, d_fu1, 1);
cublasSaxpy(N, -1.0, d_u, 1, d_fu1, 1);
```

✓
```
CVectorCUDA<dataTYPE> d_fu1(N);
d_fu1 = d_x;
d_fu1 -= d_u;
```

- If it's **too hard** or not good for CUDA, just **rip** it off the **GPU** and do it **CPU**!

CSIP

What Does a Bit of Code Look Like?



MATLAB

```
rpri = A*x – b;
```

boost C++ LIBRARIES

```
VECTOR<dataTYPE> rpri = prod(A,x) – b;
```

NVIDIA GeForce CUDA boost

CUDA + Boost

```
CVectorCUDA<dataTYPE> d_rpri(M);
d_rpri = d_b;
d_rpri *= -1;
d_rpri.add(d_A, d_x);

debugCheck(d_rpri.compare(rpri));
```

CSIP

References

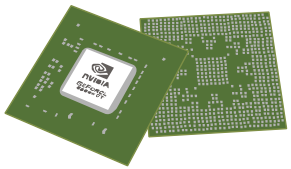
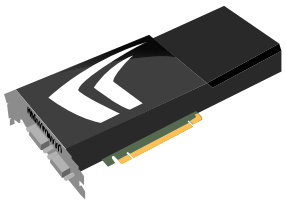
- Boost C++ Libraries.
- Fatica, Massimiliano and Jeong, Won-Ki. "Accelerating MATLAB with CUDA."
- NVIDIA Corporation. CUDA Zone.
- Jonathan Richard Shewchuk. An Introduction to the Conjugate Gradient Method Without the Agonizing Pain. August 1994.
- Romberg, Justin. "l1-MAGIC". Accessed December 10, 2008.

Questions

The End.

Backup Slides

Comparison of CPU and GPU

Attribute	Laptop System	Desktop System
Manufacturer	Dell	Dell
Model	XPS M1530	Inspiron 530
		
Processor	Intel Core 2 Duo 2.5Gz	Intel Core 2 Duo 2.2Gz
Memory	4GB	2GB
OS	Windows XP Profession 2002, SP3	Windows XP Profession 5.1, SP2
Graphics Card	**GeForce 8600M GT**	**GeForce GTX 260**
		
Driver	6.14.0011.7884 (8/7/2008)	6.14.0011.7813 (9/17/2008)
Processor Cores	16	192
Multiprocessors	2	24
Processor Clock	450 MHz	1242 MHz
Memory Clock	600 MHz	999 MHz
Memory Size	512MB	896 MB
Compute Capability	1.1	1.3