CUDA Parallel Computing Architecture

GPU Computing Applications

C++ with CUDA extensions

C with CUDA extensions

OpenCL™

Direct Compute

Fortran

Java and Python

NVIDIA GPU with the CUDA Parallel Computing Architecture

OpenCL is trademark of Apple Inc. used under license to the Khronos Group Inc.
CUDA Parallel Paradigm

- Scale to 100s of cores, 1000s of parallel threads
  - Transparently with one source and same binary

- Let programmers focus on parallel algorithms
  - Not mechanics of a parallel programming language

- Enable CPU+GPU Co-Processing
  - CPU & GPU are separate devices with separate memories
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];
}

// Invoke serial SAXPY kernel
saxpy_serial(n, 2.0, x, y);

__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];
}

// Invoke parallel SAXPY kernel with 256 threads/block
int nbblocks = (n + 255) / 256;
saxpy_parallel<<<nbblocks, 256>>>(n, 2.0, x, y);
CUDA Programming Effort / Performance

Source: MIT CUDA Course 6.963
Compiling C with CUDA Applications

```c
void serial_function(... ) {
    ...
}
void other_function(int ... ) {
    ...
}
void saxpy_serial(float ... ) {
    for (int i = 0; i < n; ++i)
        y[i] = a*x[i] + y[i];
}
void main( ) {
    float x;
    saxpy_serial(..);
    ...
}
```

Key Kernels

NVCC

(Open64)

CPU Compiler

CUDA object files

Rest of C Application

CPU object files

Linker

CPU-GPU Executable

Modify into Parallel CUDA code
Compiling CUDA

C/C++ CUDA Application

NVCC

PTX Code

PTX to Target Compiler

G80

... GPU

Target code

CPU Code
Compiling CUDA

C/C++ CUDA Application → NVCC → PTX Code → PTX to Target Compiler → G80, ..., GPU → Target code
NVCC & PTX Virtual Machine

C/C++ CUDA Application

EDG

Open64

PTX Code

float4 me = gx[gtid];
me.x += me.y * me.z;

EDG
- Separate GPU vs. CPU code

Open64
- Generates GPU PTX assembly
- Parallel Thread eXecution (PTX)
  - Virtual Machine and ISA
  - Programming model
  - Execution resources and state

ld.global.v4.f32  {f1,f3,f5,f7}, [$r9+0];
mad.f32          $f1, $f5, $f3, $f1;
GPU Computing Software Libraries and Engines

- **GPU Computing Applications**
- **Application Acceleration Engines (AXEs)**
  - SceniX, CompleX, Optix, PhysX
- **Foundation Libraries**
  - CUBLAS, CUFFT, CULA, NVCUVID/VENC, NVPP, Magma
- **Development Environment**
  - C, C++, Fortran, Python, Java, OpenCL, Direct Compute, ...

**CUDA Compute Architecture**
Fortran

- PGI Accelerator Compiler
  - For Fortran and C
  - Uses compiler directives

- NOAA F2C-ACC
  - Converts Fortran codes to CUDA C
  - Some hand-optimization expected

- FLAGON
  - Fortran 95 Library for GPU Numerics
  - Includes support for cuBLAS, cuFFT, CUDPP, etc.
C with CUDA 3.0

**Unified addressing for C and C++ pointers**
- Global, shared, local addresses
- Enables 3rd party GPU callable libraries, dynamic linking
- One 40-bit address space for load/store instructions

**Compiling for native 64-bit addressing**

**IEEE 754-2008 single & double precision**
- C99 math.h support

**Concurrent Kernels**
Python + CUDA = PyCUDA

- CUDA C Code = Strings
- Generate Code Easily
  - Automated Tuning
- Batteries included:
  GPU Arrays, RNG, ...
- Integration: numpy arrays, Plotting, Optimization, ...

- All of CUDA in a modern scripting language
- Full Documentation
- Free, open source (MIT)
- Also: PyOpenCL

http://mathema.tician.de/software/pycuda
Using jCUDA you can create cross-platform CUDA solutions, that can run on any operating system supported by CUDA without changing your code. Either select between Windows XP or Vista by Microsoft or even Linux/MacOS/Solaris systems. Current support is for both 32 and 64 bits of every platform.

**Features**

- Double precision
- Object model for CUDA programming
- CUDA 2.1 Driver API
- CUDA 2.1 Runtime API
- CUFFT routines
- OpenGL interoperability
- * Support for CUBLAS routines will be added in the future

**Operating System Support**

- Microsoft Windows
- Linux
- * Support for Mac OSX will be added in the future
- * Support for Solaris 10 (x86) will be added in the future

Courtesy of Company for Advanced Supercomputing Solutions, Ltd.

CUDA.NET

2.3.6, 14/9/2009
Updates to native wrappers, added SizeT structure to handle 32/64 systems compatibility for functions taking size_t as parameter.
Support for CUDA 2.3 through .NET bindings to CUDA functions.
Currently supported on Windows, Linux and MacOS platforms.

Features

- Double precision
- Object model for CUDA programming
- CUDA 2.2 Driver API
- CUDA 2.2 Runtime API
- CUFFT routines
- CUBLAS routines
- Direct3D interoperability
- OpenGL interoperability

Operating System Support

- Microsoft Windows
- Linux 32/64 bit (using Mono)
- Mac OSX (using Mono)

Courtesy of Company for Advanced Supercomputing Solutions, Ltd.

OpenCL.NET

1.0.43.2, 14/9/2009
Release 1.0.43.2 of OpenCL.NET conforms to OpenCL version 1.0.43.
(Support for OpenGL interoperability was changed in this version by Khronos)
Added SizeT structure to better handle 32/64 systems support with functions taking size_t as parameter.
Support for OpenCL 1.0.43 through .NET bindings to CUDA functions.
Currently supported on Windows, Linux and MacOS platforms.

Features

- OpenCL interface
- Object model for OpenCL programming
- OpenGL interoperability

Operating System Support

- Microsoft Windows
- Linux (using Mono)
- Mac OSX (using Mono)

Courtesy of Company for Advanced Supercomputing Solutions, Ltd.

<table>
<thead>
<tr>
<th>Solution</th>
<th>Approach</th>
<th>Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>CUDA C + Runtime</td>
<td>Language Integration</td>
<td>NVIDIA CUDA Toolkit</td>
</tr>
<tr>
<td>Fortran</td>
<td>Auto Parallelization</td>
<td>PGI Accelerator</td>
</tr>
<tr>
<td>OpenCL</td>
<td>Device-Level API</td>
<td>Khronos standard</td>
</tr>
<tr>
<td>DirectCompute</td>
<td>Device-Level API</td>
<td>Microsoft</td>
</tr>
<tr>
<td>PyCUDA</td>
<td>API Bindings</td>
<td>Open source</td>
</tr>
<tr>
<td>jCUDA</td>
<td>API Bindings</td>
<td>Freely Available</td>
</tr>
<tr>
<td>CUDA.NET</td>
<td>API Bindings</td>
<td>Freely Available</td>
</tr>
<tr>
<td>OpenCL.NET</td>
<td>API Bindings</td>
<td>Freely Available</td>
</tr>
</tbody>
</table>