Parallel Computing with MATLAB

Advanced Big Data Anayltics

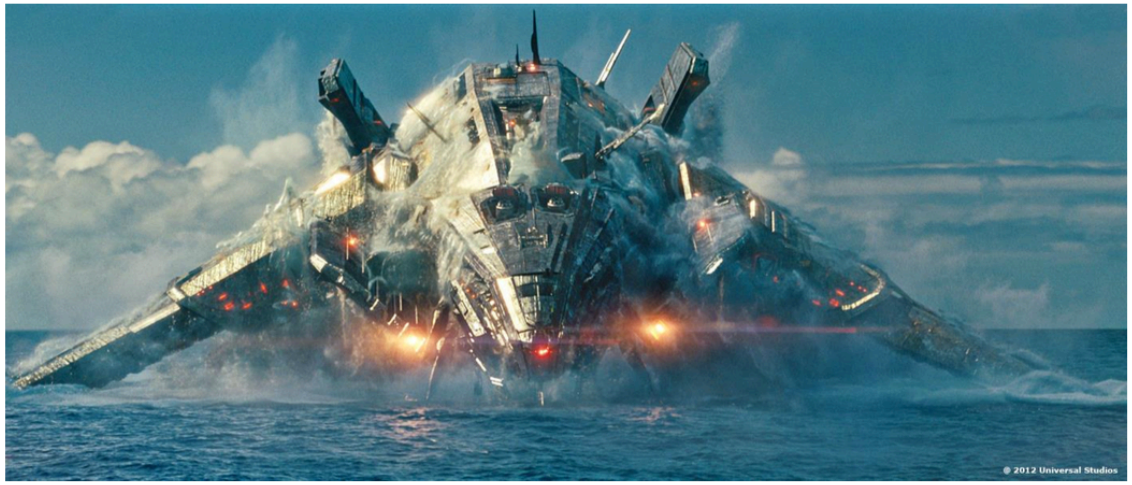
GPU in Graphics

Hollywood Visual Effects

One often cannot film various real-world situations required in order to tell a story

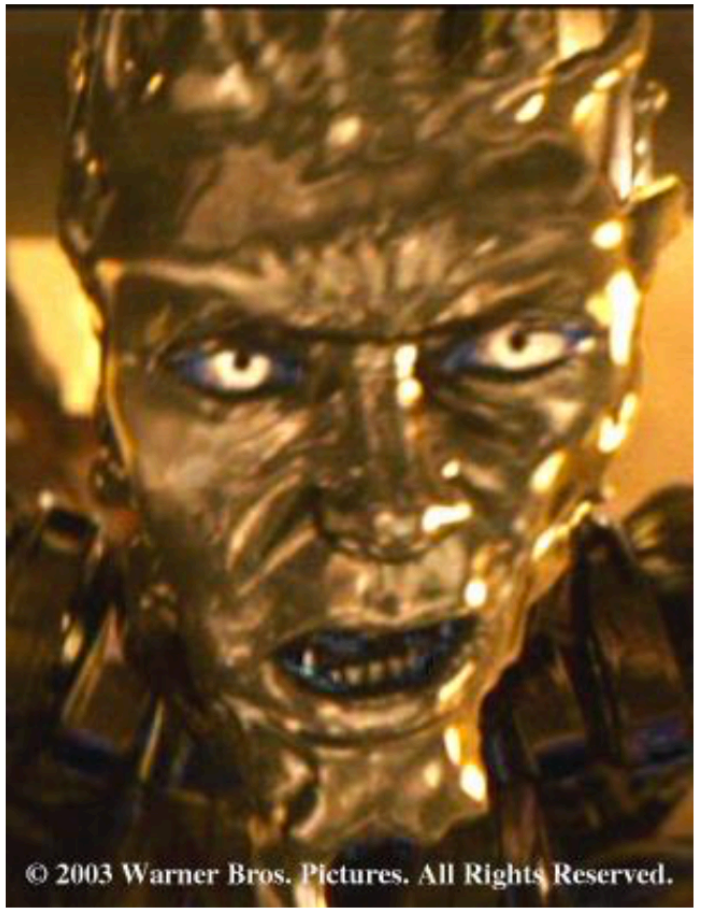
- Some times the situation is too dangerous, impractical, expensive, or rare
- Other times the situation doesn't actually exist, except in an alternative reality

Visual Effects: Liquids



Battleship

The Day After Tomorrow



Terminator 2

Visual Effects: Gases

Harry Potter and the Order of the Phoenix

Terminator 3

Star Wars Episode III

SIGGRAPH 2015 http://kesen.realtimerendering.com/sig2015.html

SIGGRAPH 2016 http://kesen.realtimerendering.com/sig2016.html

Siggraph Asia
Siggraph
IEEE IEEE Transactions on Visualization and Computer Graphics(TCVG)
Chinagraph

Visual Effects: Solids

Destruction:
fracture,
explosions,
etc.

Super 8

2012

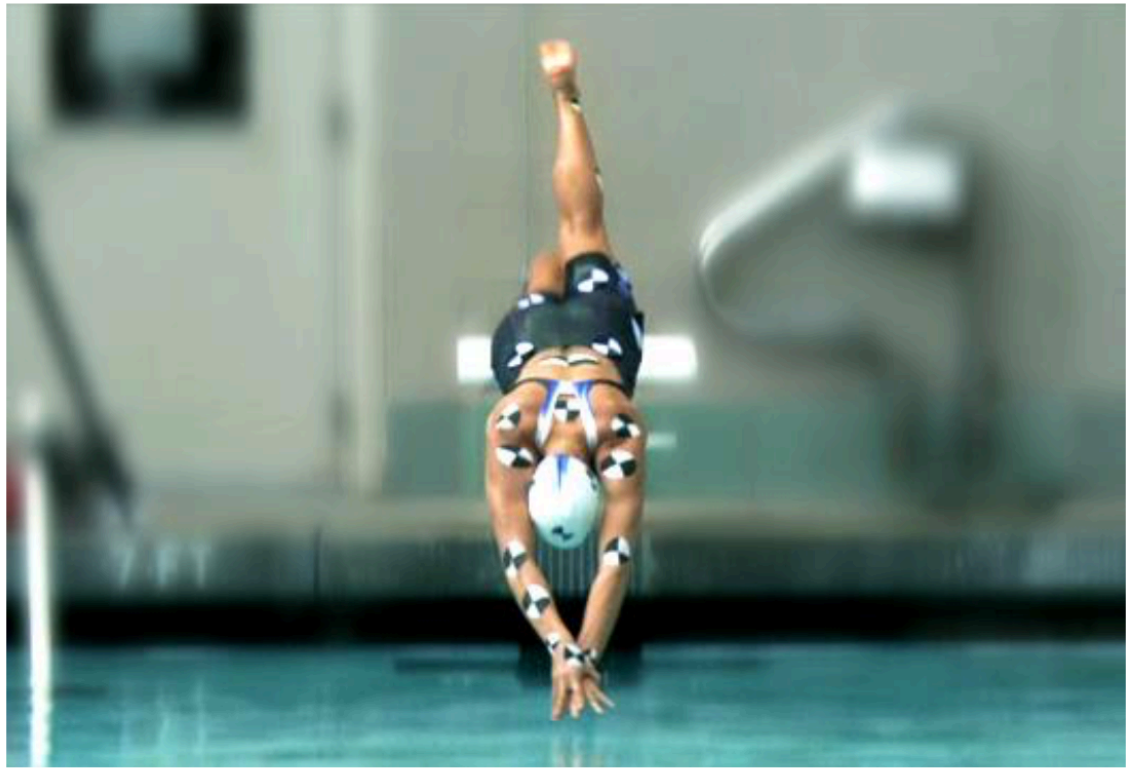
Visual Effects: CG Creatures

Yoda, Star Wars Episode II

Sméagol/Gollum, The Lord of the Rings

Motion Capture

Facial capture in Avatar



Motion capture of Olympic swimmer Dana Vollmer by Manhattan Mocap
(technology transition)

Animated Films

Toy Story 3

Monsters, Inc.

Video Games

Spore

Crysis

Braid

Virtual (and Augmented) Reality

Ivan Sutherland: Head-mounted displays, with mechanical tracker

Oculus Rift

Scientists/Engineers need graphics too

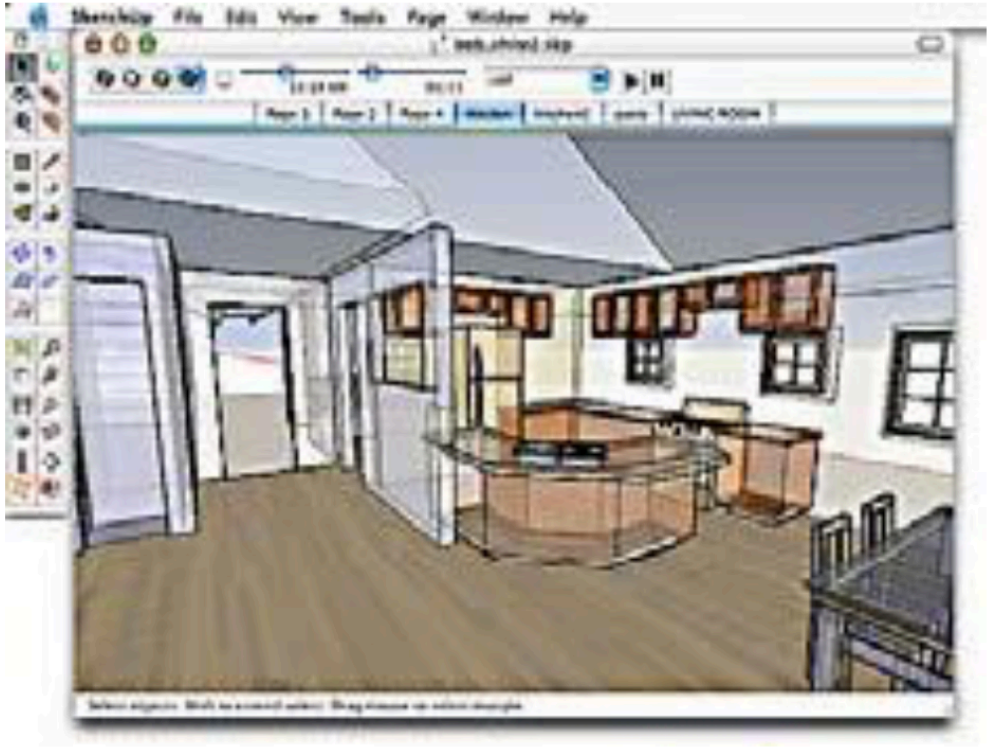
- Visualization of various phenomena, computer aided design, virtual prototyping, simulation, etc.
- Learning at least a little bit about a graphics is highly useful for most scientists and engineers as well...

Scientific Visualization

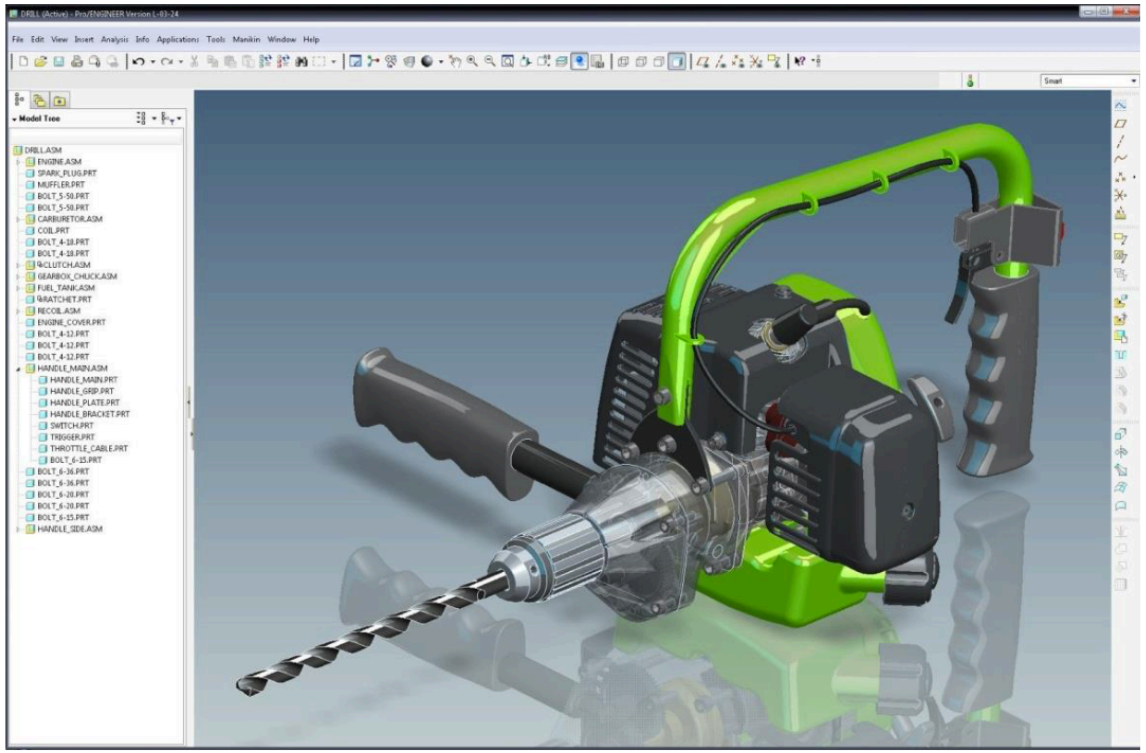
The Virtual Human
Karl-Heinz Hoehne

Outside-In
The Geometry Center

Computer-Aided Design



Sketchup



ProEngineer

Visual Simulation and Training

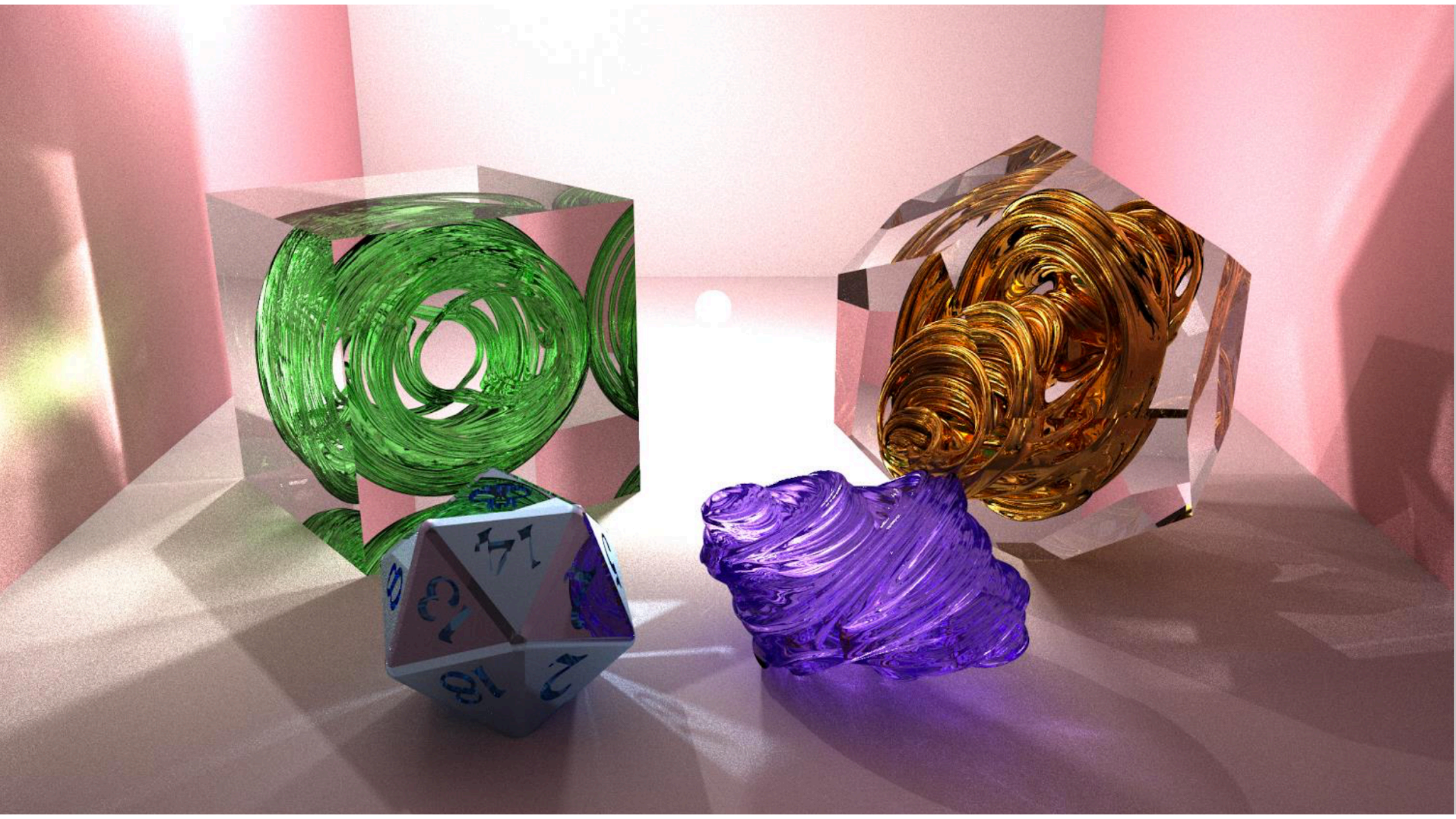
- Apollo spacecraft
- Flight simulators
- Driving simulators
- Surgical simulation

Davinci surgical robot
Intuitive Surgical

Driving simulator
Toyota Higashifuji Technical Center

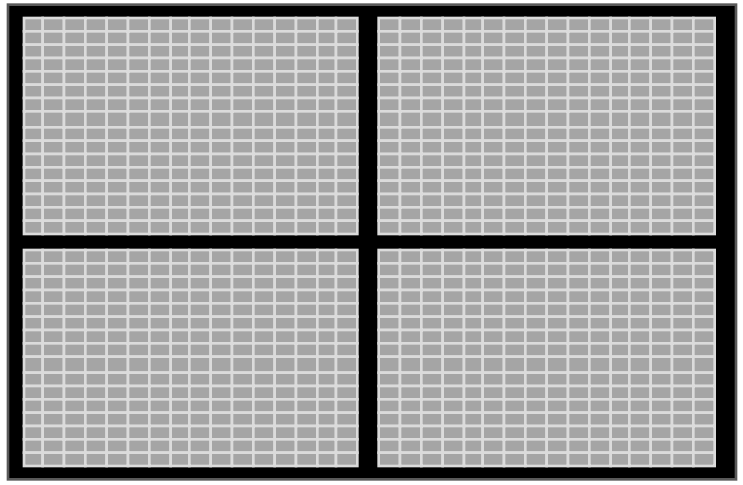
More Effects

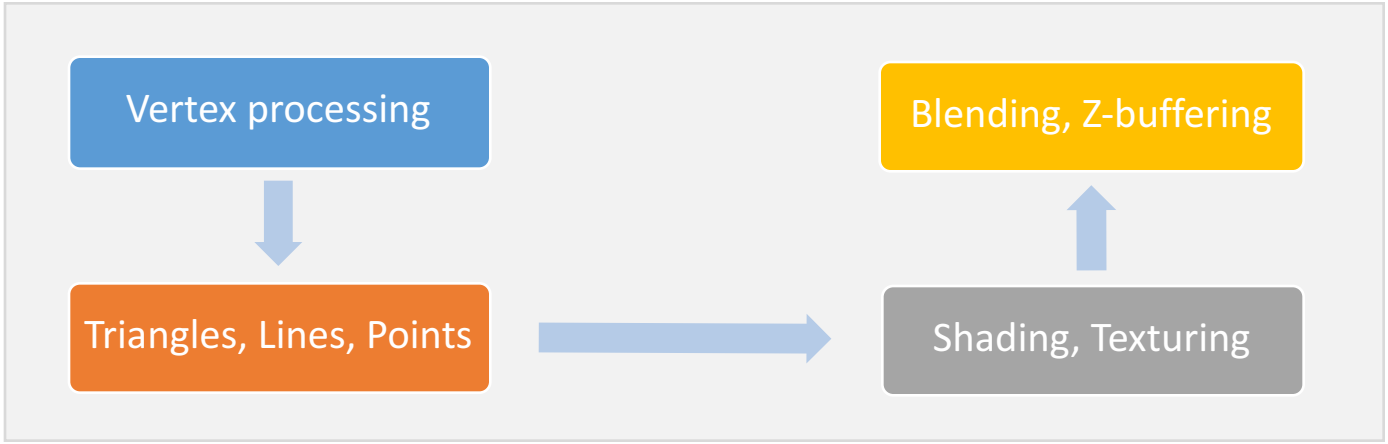

成功
培養人一己百的
克勤精神剛天下
斷無不成功之事




Ship in Bottle
$3248

Traditional GPU workflow





GPGPU

1999-2000 computer scientists from various fields started using GPUs to accelerate a range of scientific applications.

GPU programming required the use of graphics APIs such as OpenGL and Cg.

2002 James Fung (University of Toronto) developed OpenVIDIA.

NVIDIA greatly invested in GPGPU movement and offered a number of options and libraries for a seamless experience for C, C++ and Fortran programmers.

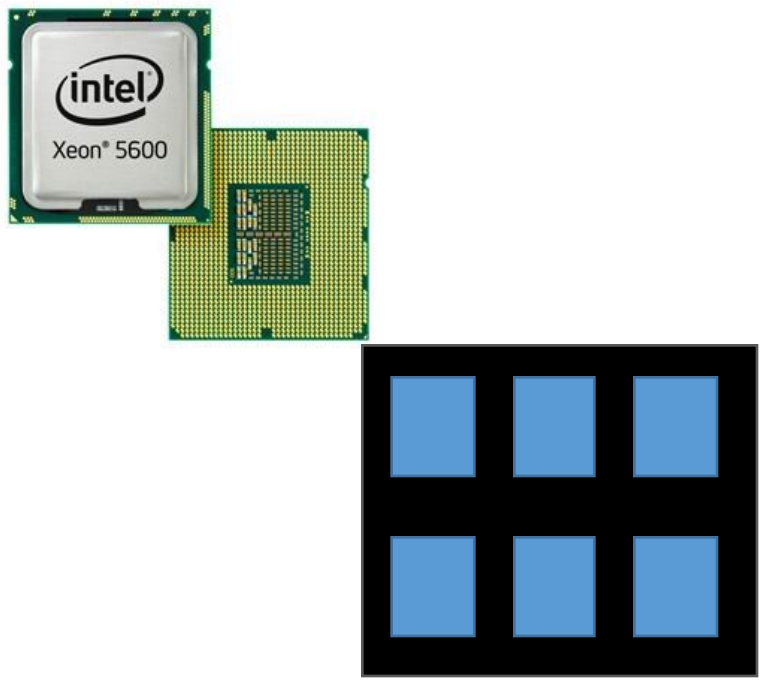
GPGPU timeline

In November 2006 Nvidia launched CUDA, an API that allows to code algorithms for execution on Geforce GPUs using C programming language.

Khronus Group defined OpenCL in 2008 supported on AMD, Nvidia and ARM platforms.

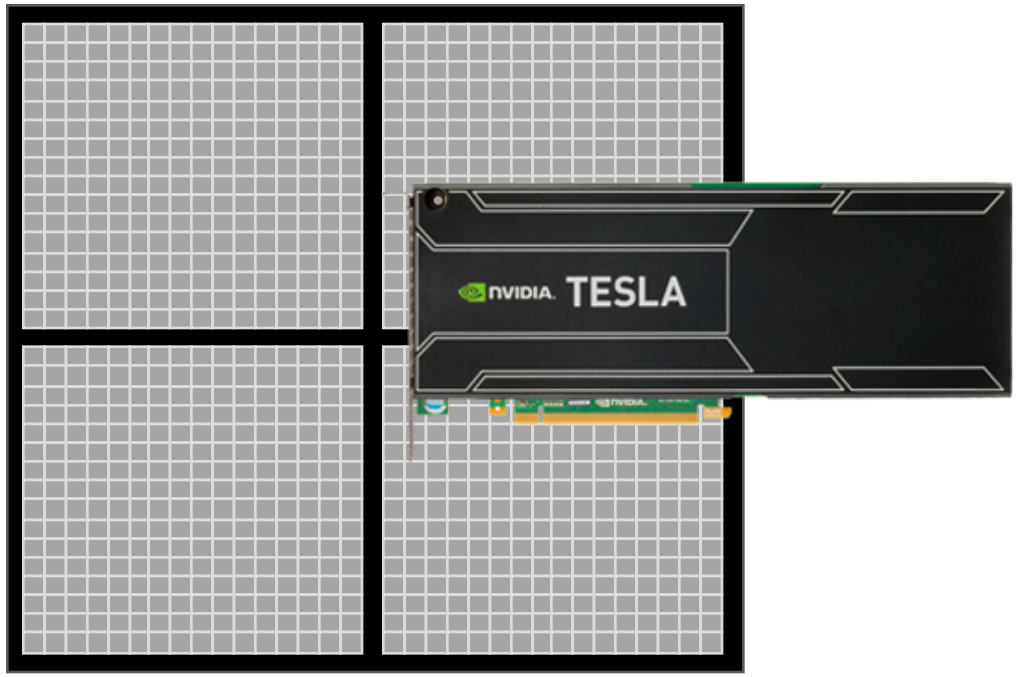
In 2012 Nvidia presented and demonstrated OpenACC - a set of directives that greatly simplify parallel programming of heterogeneous systems.

CPU



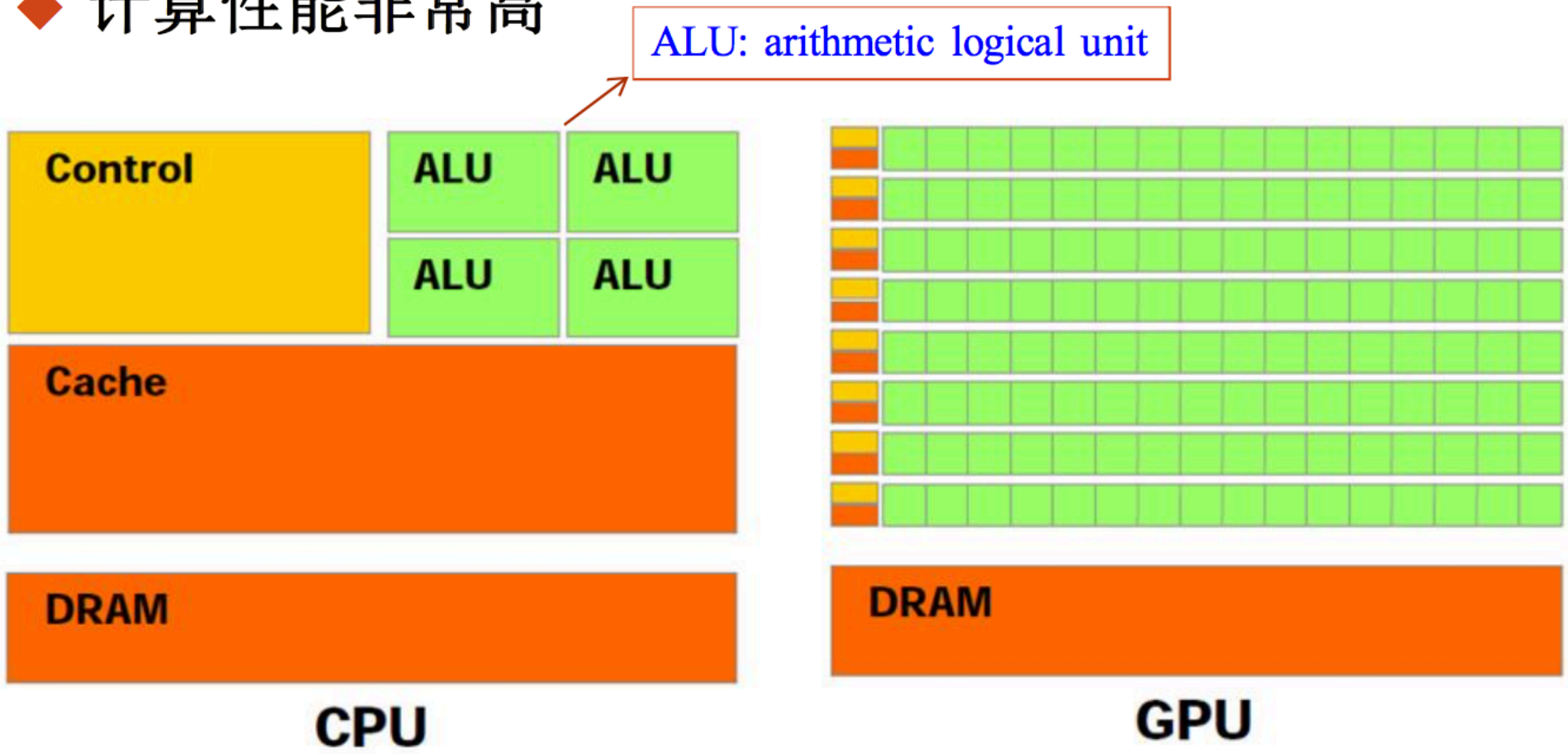
CPUs consist of a few cores optimized for serial processing

GPU



GPUs consist of hundreds or thousands of smaller, efficient cores designed for parallel performance

◆ 计算性能非常高



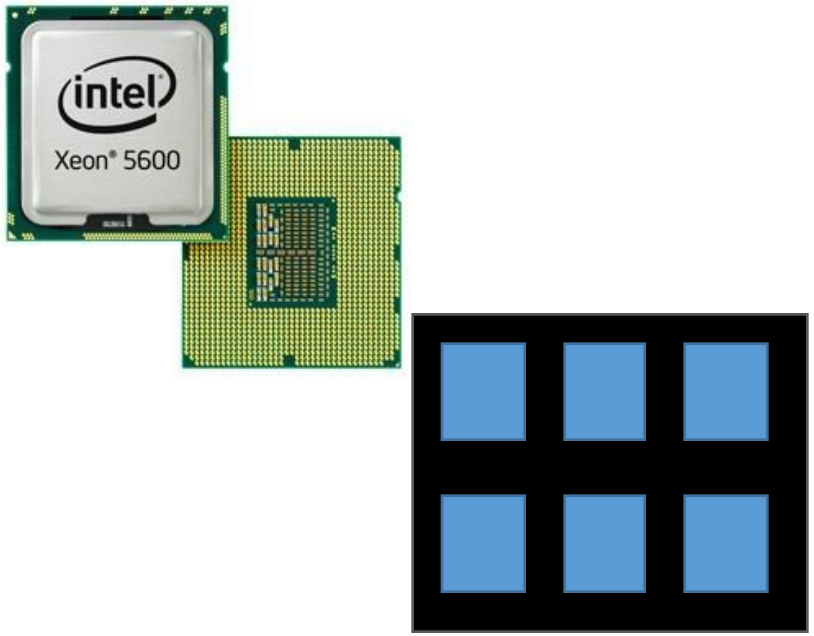
更多资源用于缓存

更多资源用于数据计算

Core i7： 4 cores 100GFlops

TeslaK20 > 3.95TFlops

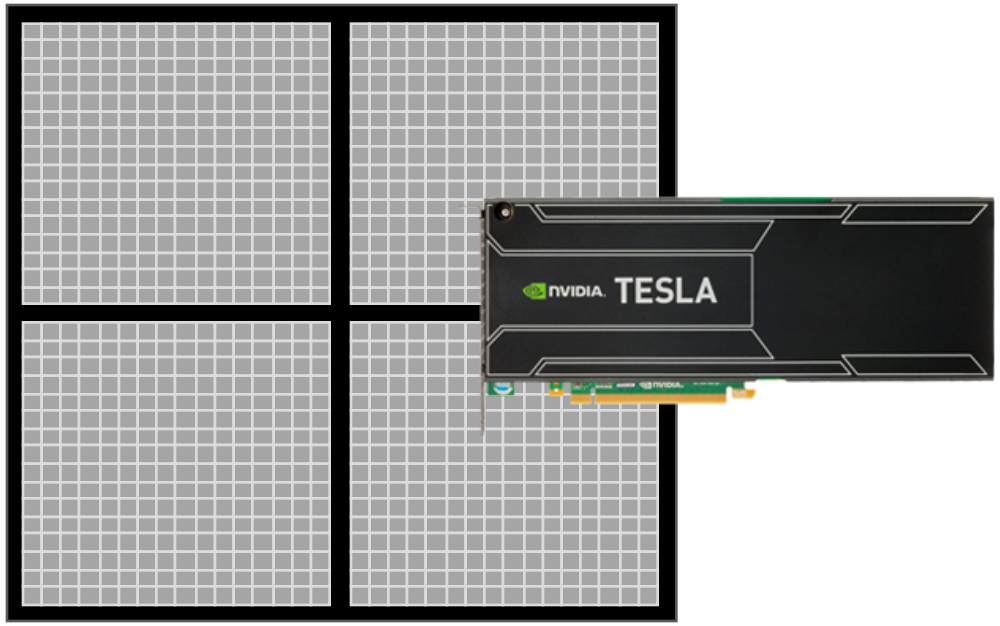
SCC CPU



Intel Xeon X5650:

Memory size: **288** GB
Bandwidth: **32** GB/sec

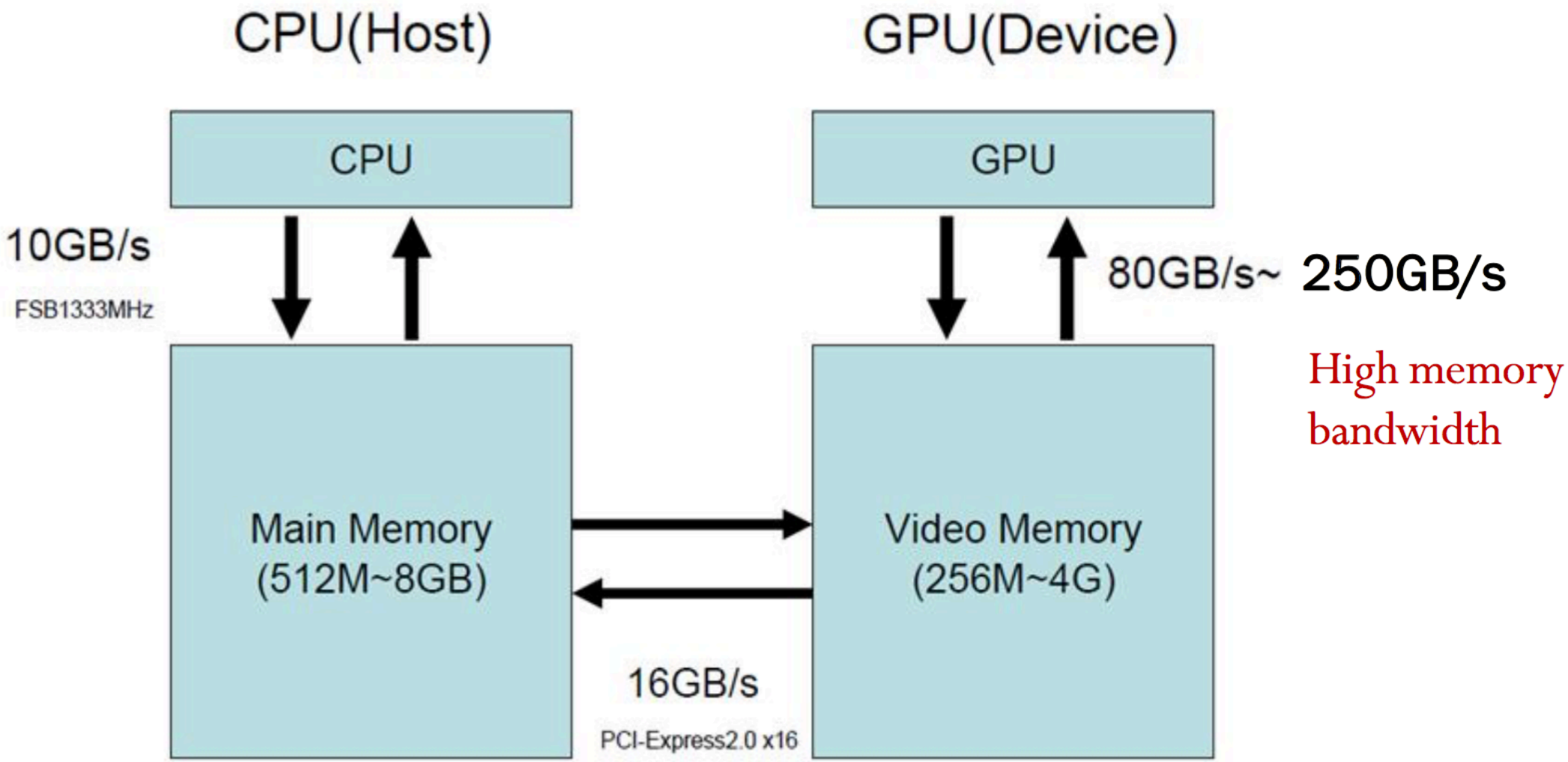
SCC GPU



NVIDIA Tesla M2070:

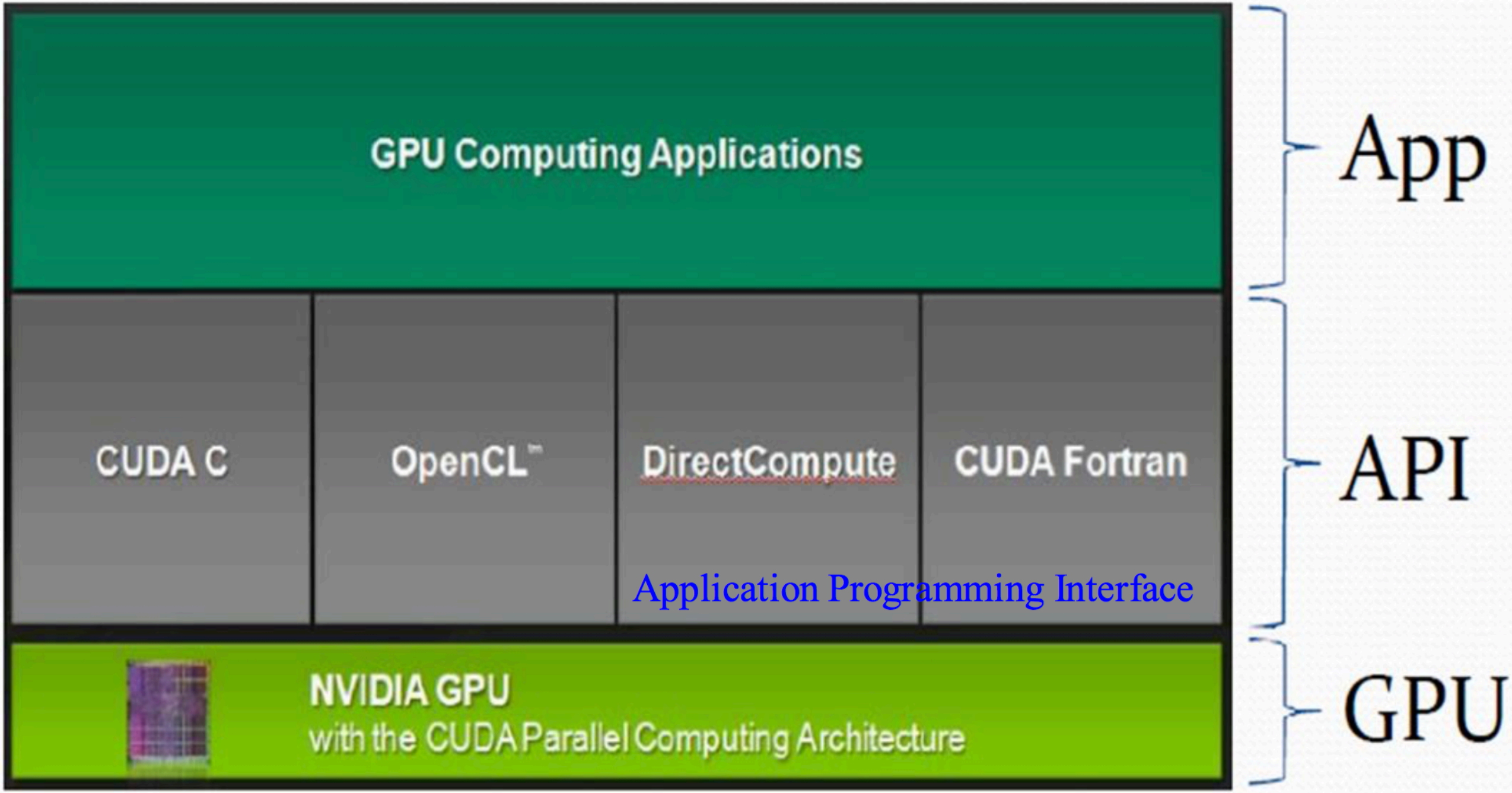
Memory size: **3GB** total
Bandwidth: **150** GB/sec

带宽非常高


CPU(Host)
GPU(Device)
CPU
GPU
10GB/s
FSB1333MHz
80GB/s~ 250GB/s
High memory bandwidth
Main Memory
(512M~8GB)
Video Memory
(256M~4G)
16GB/s
PCI-Express2.0 x16

GPU Computing Growth

2008		2013
100M CUDA-capable GPUs	x 4.3	430M CUDA-capable GPUs
150K CUDA downloads	x 10.67	1.6M CUDA downloads
1 Supercomputer	x 50	50 Supercomputers
4,000 Academic Papers	x 9.25	37,000 Academic Papers


GPU Computing Applications
CUDA C
OpenCL™
DirectCompute
CUDA Fortran
Application Programming Interface
NVIDIA GPU
with the CUDA Parallel Computing Architecture
App
API
GPU

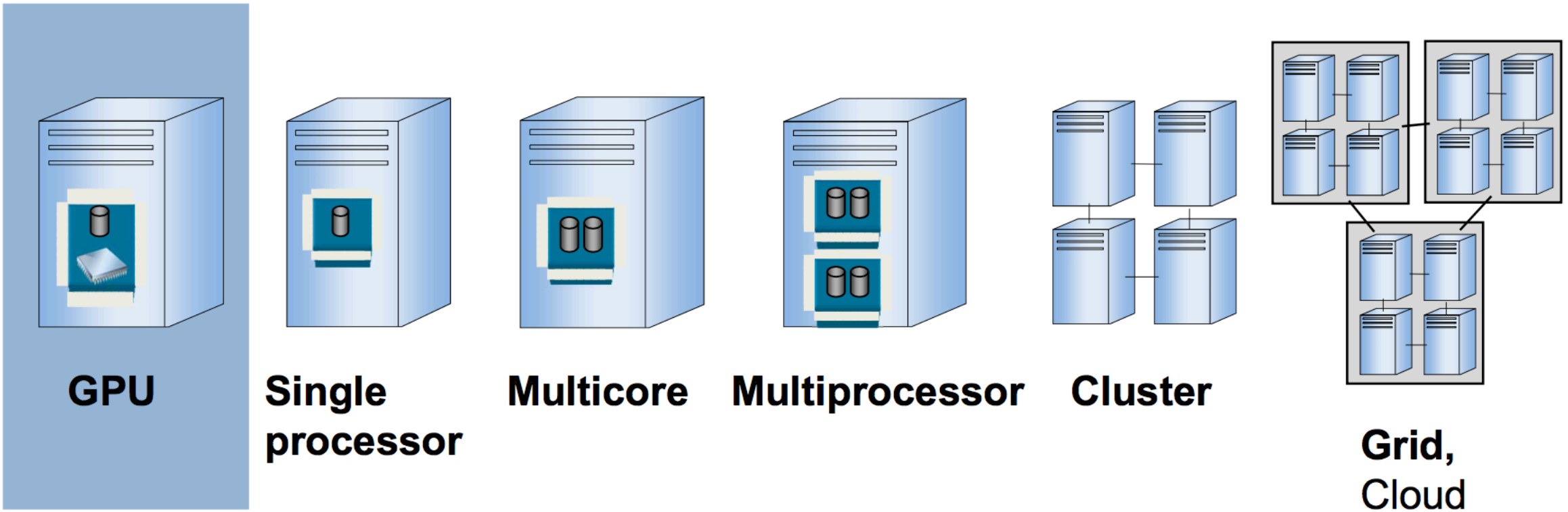
MATLAB

Parallel toolbox

Parallel Capabilities

Task Parallel	Data Parallel	Environment
Built-in support with Simulink, toolboxes, and blocksets		`matlabpool` Local workers
		Configurations
`parfor`	distributed array >200 functions	`batch` MathWorks job manager
`job/task`	`spmd` co-distributed array MPI interface	third-party schedulers `job/task`

Evolving With Technology Changes



GPU

Single processor

Multicore

Multiprocessor

Cluster

Grid, Cloud

GPU Arrays

```
>> A = someArray(1000, 1000);
>> G = gpuArray(A); % Push to GPU memory
…
>> F = fft(G);
>> x = G\b;
…
>> z = gather(x); % Bring back into MATLAB
```

Spectrogram on the desktop
(CPU only)

```
D = data;
iterations = 2000; % # of parallel iterations
stride = iterations*step; %stride of outer loop

M = ceil((numel(x)-W)/stride);%iterations needed
o = cell(M, 1); % preallocate output

for i = 1:M
    % What are the start points
    thisSP = (i-1)*stride:step: …
        (min(numel(x)-W, i*stride)-1);

    % Move the data efficiently into a matrix
    X = copyAndWindowInput(D, window, thisSP);

    % Take lots of fft's down the colmuns
    X = abs(fft(X));

    % Return only the first part to MATLAB
    o{i} = X(1:E, 1:ratio:end);

end
```

Read my codes:
Fu_VQA_main_iteratively_W1_W2_version2_GPU.m

Spectrogram on the desktop
(CPU to GPU)

```
D = data;
iterations = 2000; % # of parallel iterations
stride = iterations*step; %stride of outer loop

M = ceil((numel(x)-W)/stride);%iterations needed
o = cell(M, 1); % preallocate output

for i = 1:M
    % What are the start points
    thisSP = (i-1)*stride:step: …
        (min(numel(x)-W, i*stride)-1);

    % Move the data efficiently into a matrix
    X = copyAndWindowInput(D, window, thisSP);

    % Take lots of fft's down the colmuns
    X = abs(fft(X));

    % Return only the first part to MATLAB
    o{i} = X(1:E, 1:ratio:end);

end
```

```
D = gpuArray(data);
iterations = 2000; % # of parallel iterations
stride = iterations*step; %stride of outer loop

M = ceil((numel(x)-W)/stride);%iterations needed
o = cell(M, 1); % preallocate output

for i = 1:M
    % What are the start points
    thisSP = (i-1)*stride:step: ...
        (min(numel(D)-W, i*stride)-1);

    % Move the data efficiently into a matrix
    X = copyAndWindowInput(D, window, thisSP);

    % Take lots of fft's down the colmuns
    X = gather(abs(fft(X)));

    % Return only the first part to MATLAB
    o{i} = X(1:E, 1:ratio:end);

end
```

Spectrogram on the desktop
(GPU to parallel GPU)

```
D = gpuArray(data);
iterations = 2000; % # of parallel iterations
stride = iterations*step; %stride of outer loop

M = ceil((numel(x)-W)/stride);%iterations needed
o = cell(M, 1); % preallocate output

for i = 1:M
    % What are the start points
    thisSP = (i-1)*stride:step: ...
        (min(numel(D)-W, i*stride)-1);

    % Move the data efficiently into a matrix
    X = copyAndWindowInput(D, window, thisSP);

    % Take lots of fft's down the colmuns
    X = gather(abs(fft(X)));

    % Return only the first part to MATLAB
    o{i} = X(1:E, 1:ratio:end);

end
```

```
D = gpuArray(data);
iterations = 2000; % # of parallel iterations
stride = iterations*step; %stride of outer loop

M = ceil((numel(x)-W)/stride);%iterations needed
o = cell(M, 1); % preallocate output

parfor i = 1:M
    % What are the start points
    thisSP = (i-1)*stride:step: ...
        (min(numel(D)-W, i*stride)-1);

    % Move the data efficiently into a matrix
    X = copyAndWindowInput(D, window, thisSP);

    % Take lots of fft's down the colmuns
    X = gather(abs(fft(X)));

    % Return only the first part to MATLAB
    o{i} = X(1:E, 1:ratio:end);

end
```