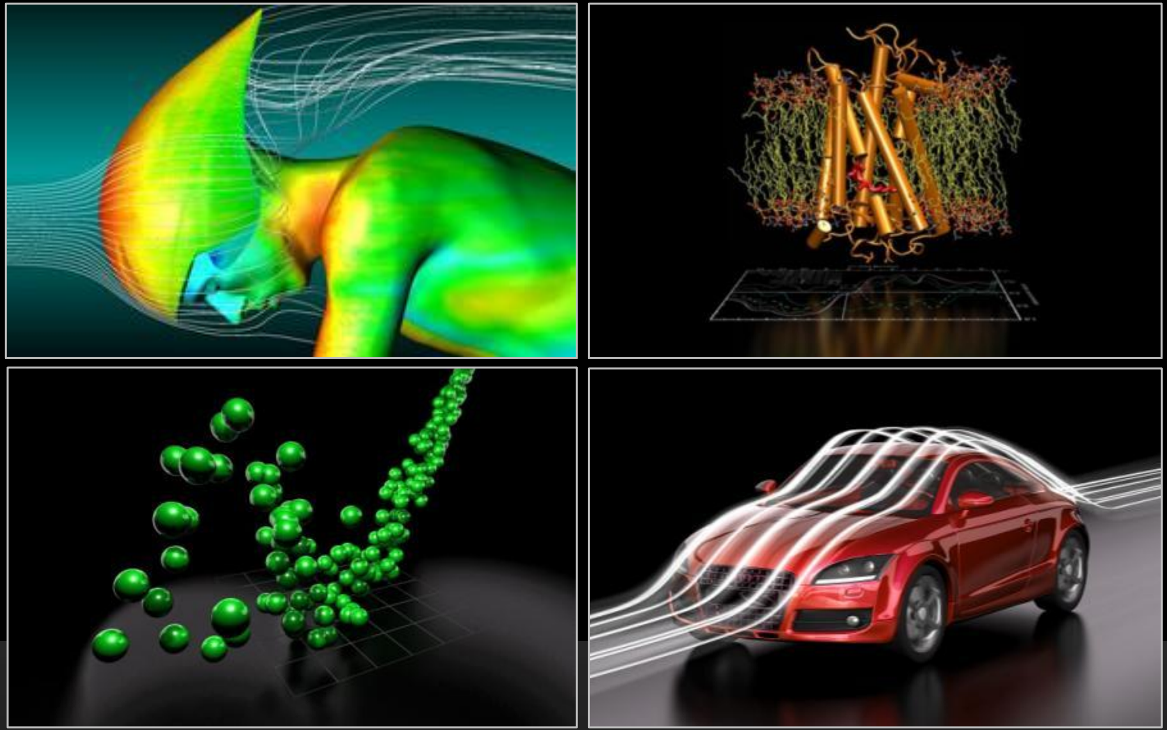
TESLA

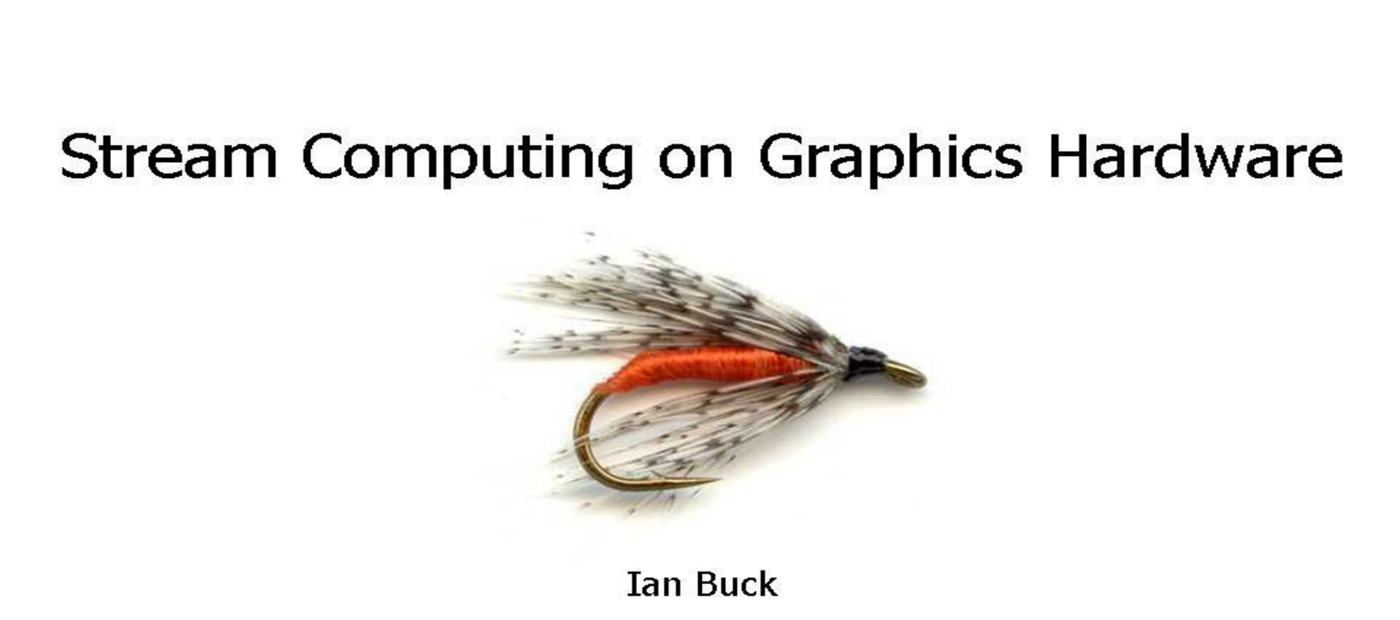
GPU Computing

Past, Present, Future



Ian Buck, GM GPU Computing Software

History....

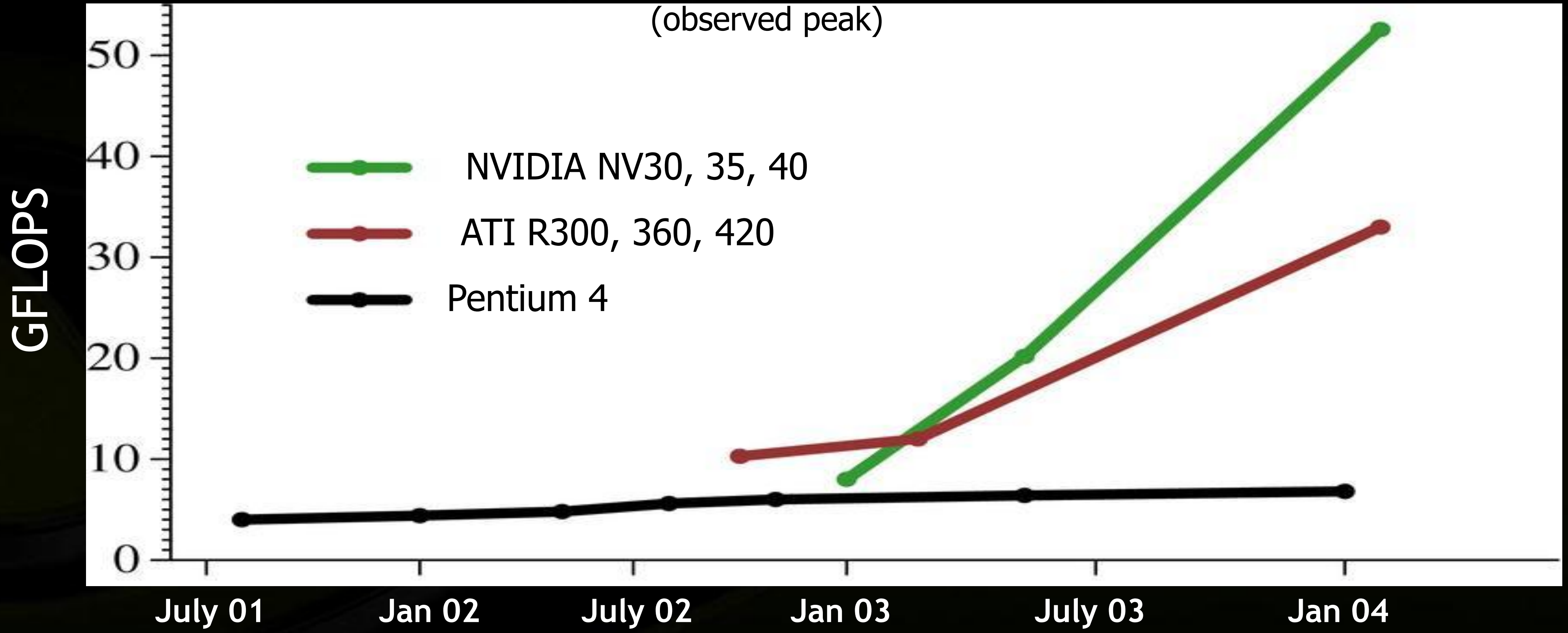


Stream Computing on Graphics Hardware

Ian Buck

GPGPU in 2004

recent trends



GPU history

	Product	Process	Trans	MHz	GFLOPS (MUL)
Aug-02	GeForce FX5800	0.13	121M	500	8
Jan-03	GeForce FX5900	0.13	130M	475	20
Dec-03	GeForce 6800	0.13	222M	400	53

translating transistors into performance

- 1.8x increase of transistors
- 20% *decrease* in clock rate
- 6.6x GFLOP speedup

Stunning Graphics Realism

Lush, Rich Worlds

Crysis © 2006 Crytek / Electronic Arts

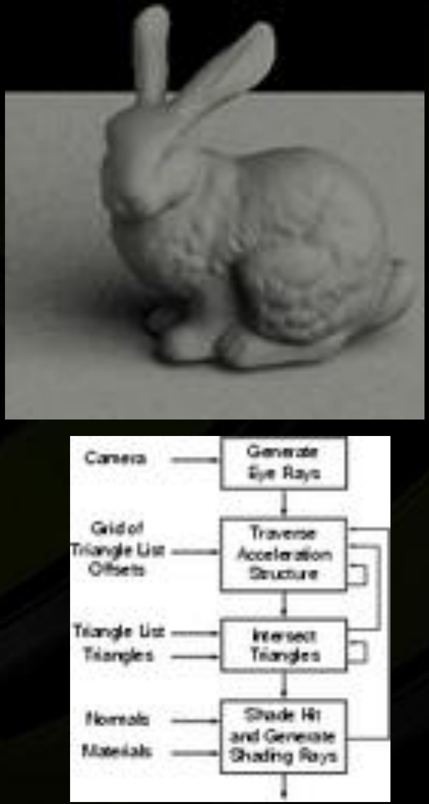
Incredible Physics Effects

Core of the Definitive Gaming Platform

Hellgate: London © 2005-2006 Flagship Studios, Inc. Licensed by NAMCO BANDAI Games America, Inc.

Full Spectrum Warrior: Ten Hammers © 2006 Pandemic Studios, LLC. All rights reserved. © 2006 THQ Inc. All rights reserved.

Early GPGPU (2002)



Camera
Generate Eye Rays
Grid of Triangle List Offsets
Traverse Acceleration Structure
Triangle List
Triangles
Intersect Triangles
Normals
Materials
Shade Hit and Generate Shading Rays

Early Raytracing

GPGPU

www.gpgpu.org

- **Ray Tracing on Programmable Graphics Hardware**
  Purcell *et al.*
- **PDEs in Graphics Hardware**
  Strzodka,,Rumpf
- **Fast Matrix Multiplies using Graphics Hardware**
  Larsen, McAllister
- **Using Modern Graphics Architectures for General-Purpose Computing: A Framework and Analysis.**
  Thompson *et al.*

Programming model challenge

- Demonstrate GPU performace
- PHD computer graphics to do this
- Financial companies hiring game programmers

- “GPU as a processor”

Brook (2003)

C with streams

- streams
  - collection of records requiring similar computation
    - particle positions, voxels, FEM cell, …

```
Ray r<200>;
float3 velocityfield<100,100,100>;
```

  - similar to arrays, but…
    - index operations disallowed: `position[i]`
    - read/write stream operators:

```
streamRead (positions, p_ptr);
streamWrite (velocityfield, v_ptr);
```

Building GPU Computing Ecosystem

- Convince the world to program an entirely new kind of processor
- Tradeoffs between functional vs. performance requirements
- Deliver HPC feature parity
- Seed larger ecosystem with foundational components

CUDA: C on the GPU

- A simple, explicit programming language solution
- Extend only where necessary

```
__global__ void KernelFunc(...);
__shared__ int SharedVar;

KernelFunc<<< 500, 128 >>>(...);
```

- Explicit GPU memory allocation
  - `cudaMalloc()`, `cudaFree()`
- Memory copy from host to device, etc.
  - `cudaMemcpy()`, `cudaMemcpy2D()`, ...

CUDA: Threading in Data Parallel

- Threading in a data parallel world
  - Operations drive execution, not data
- Users simply given thread id
  - They decide what thread access which data element
  - One thread = single data element or block or variable or nothing….
  - No need for accessors, views, or built-ins
- Flexibility
  - Not requiring the data layout to force the algorithm
  - Blocking computation for the memory hierarchy (shared)
  - Think about the algorithm, not the data

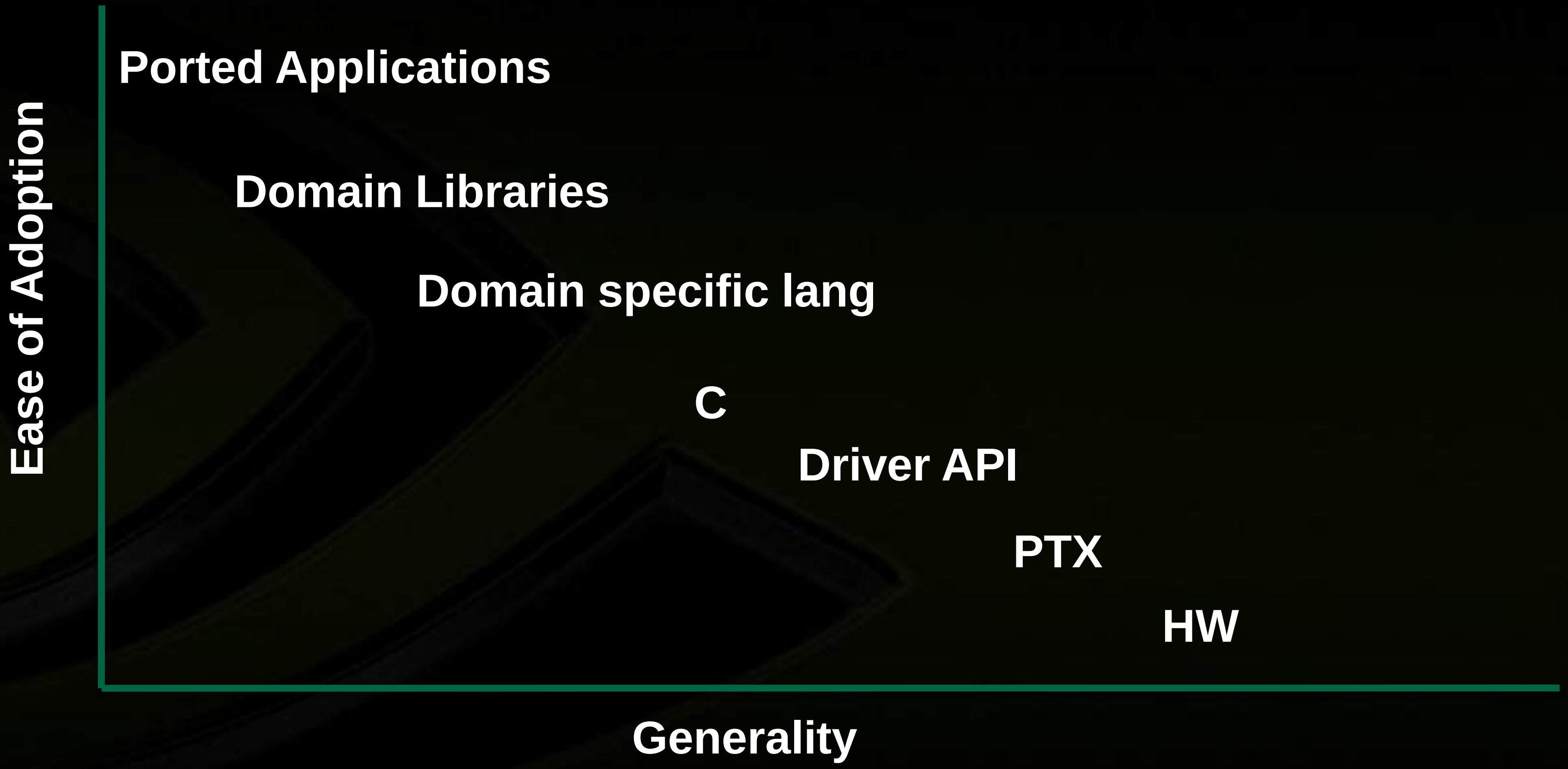
Divergence in Parallel Computing

- Removing divergence pain from parallel programming

- SIMD Pain
  - User required to SIMD-ify
  - User suffers when computation goes divergent

- GPUs: Decouple execution width from programming model
  - Threads can diverge freely
  - Inefficiency only when granularity exceeds native machine width
  - Hardware managed
  - Managing divergence becomes performance optimization
  - Scalable

Customizing Solutions



Ported Applications

Domain Libraries

Domain specific lang

C

Driver API

PTX

HW

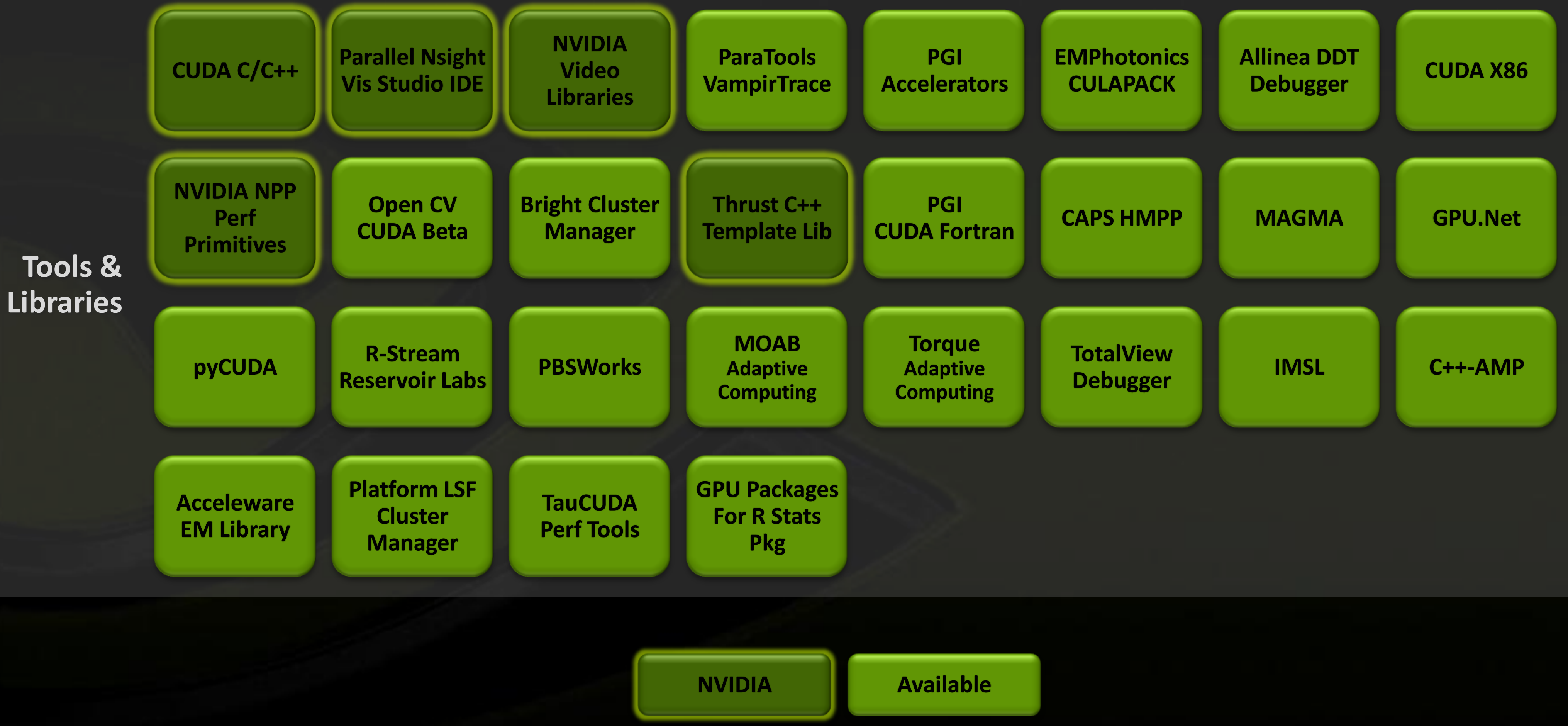
Ease of Adoption

Generality

GPU Computing By the Numbers:

>350,000,000	Compute Capable GPUs
>1,000,000	Toolkit Downloads
>120,000	Active CUDA Developers
>450	Universities Teaching CUDA
100%	OEMs offer CUDA GPU PCs

Developer ecosystem enables the application growth



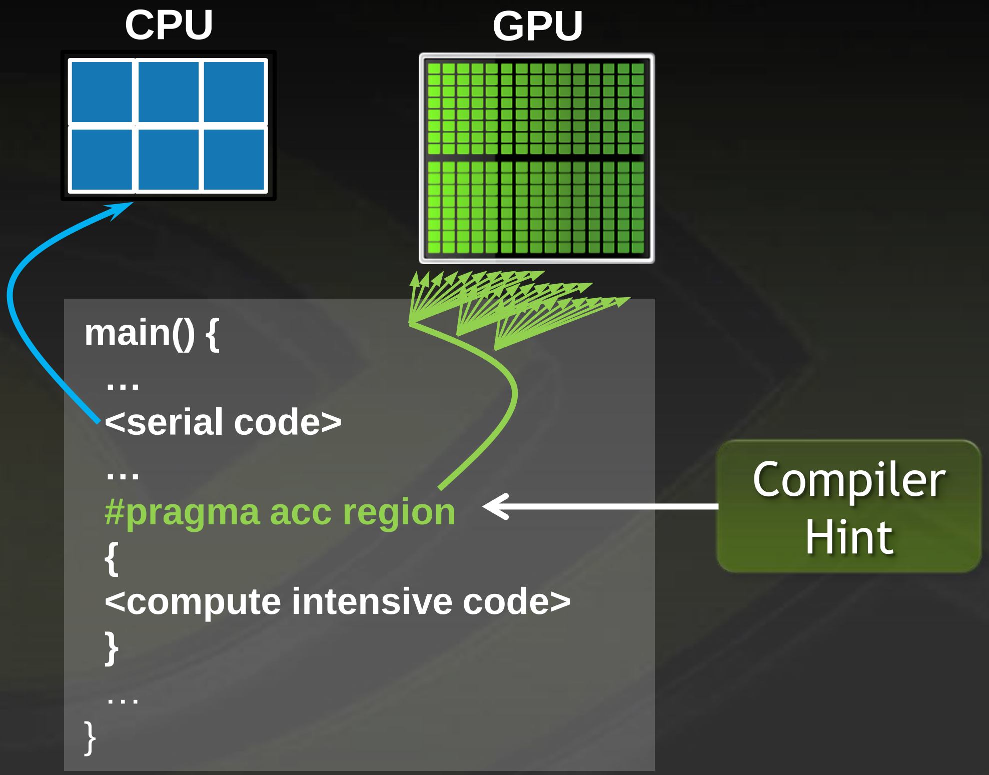
Tools & Libraries

CUDA C/C++	Parallel Nsight Vis Studio IDE	NVIDIA Video Libraries	ParaTools VampirTrace	PGI Accelerators	EMPhotonics CULAPACK	Allinea DDT Debugger	CUDA X86
NVIDIA NPP Perf Primitives	Open CV CUDA Beta	Bright Cluster Manager	Thrust C++ Template Lib	PGI CUDA Fortran	CAPS HMPP	MAGMA	GPU.Net
pyCUDA	R-Stream Reservoir Labs	PBSWorks	MOAB Adaptive Computing	Torque Adaptive Computing	TotalView Debugger	IMSL	C++-AMP
Acceleware EM Library	Platform LSF Cluster Manager	TauCUDA Perf Tools	GPU Packages For R Stats Pkg				

NVIDIA

Available

Directives: Simple Hints for the Compiler



```
main() {
  ...
  <serial code>
  ...
  #pragma acc region
  {
  <compute intensive code>
  }
  ...
}
```

Your original C/Fortran code

Add hints to code

On-ramp to parallel computing

Compiler does heavy lifting of parallelizing code

Works on multicore CPUs & many core GPUs

2x in 4 Weeks. Guaranteed.

NVIDIA®

PGI®

Free 30 day trial license
to PGI Accelerator*

Tools for quick ramp

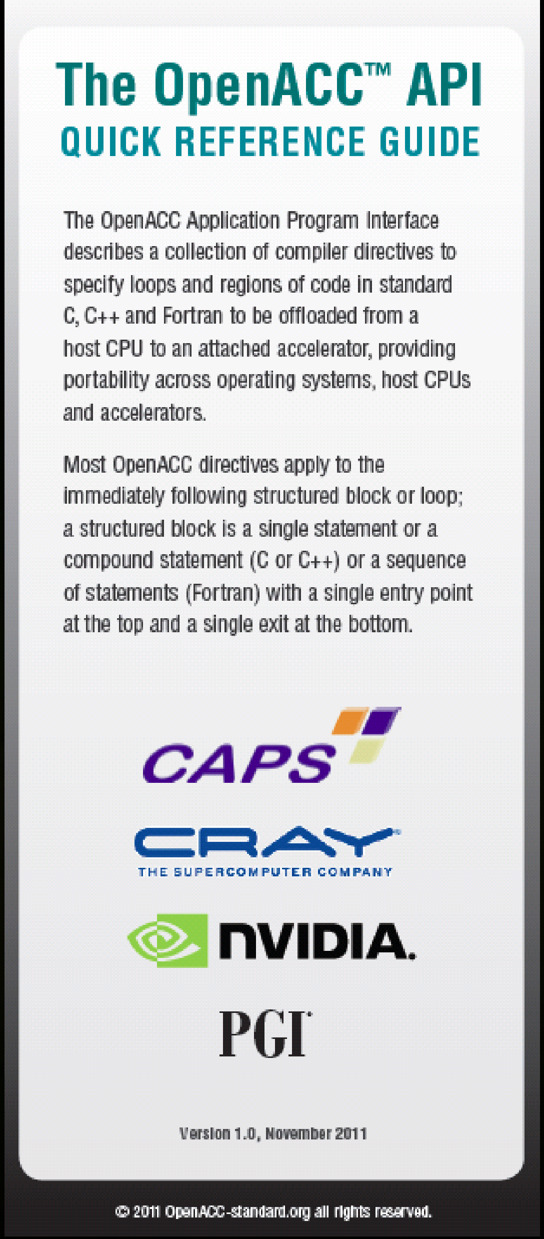
www.nvidia.com/2xin4weeks

*Limit 1000 developers

OpenACC: Open Programming Standard for Parallel Computing

Easy, Fast, Portable

http://www.openacc-standard.org



The OpenACC™ API

QUICK REFERENCE GUIDE

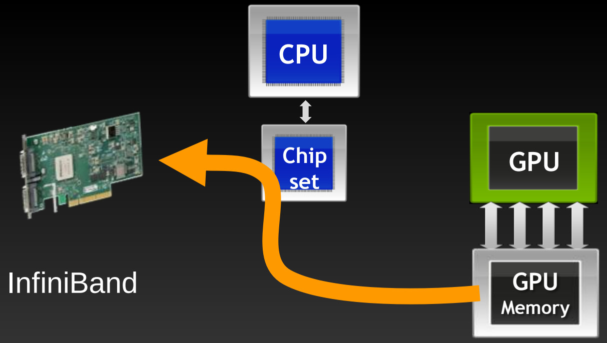
The OpenACC Application Program Interface describes a collection of compiler directives to specify loops and regions of code in standard C, C++ and Fortran to be offloaded from a host CPU to an attached accelerator, providing portability across operating systems, host CPUs and accelerators.

Most OpenACC directives apply to the immediately following structured block or loop; a structured block is a single statement or a compound statement (C or C++) or a sequence of statements (Fortran) with a single entry point at the top and a single exit at the bottom.

Version 1.0, November 2011

Building blocks for Exascale

GPU Direct

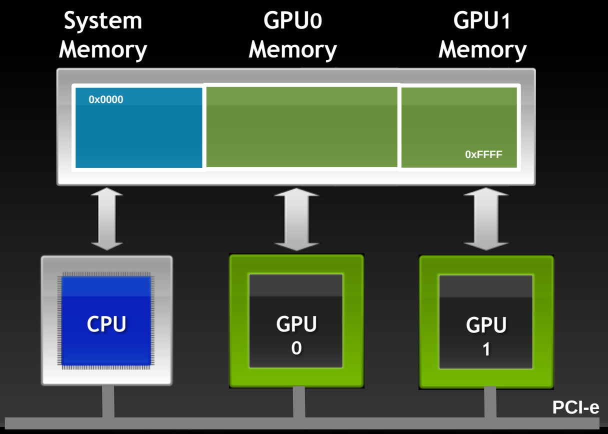


Atomic Ops

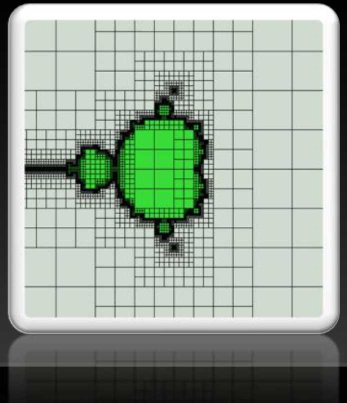
Atomic operations for thread-to-thread communication

```
atom{.space}.op.type  d, [a], b;
atom{.space}.op.type  d, [a], b, c;
.space = { .global, .shared };
.op    = { .and, .or, .xor,          // .b32 only
           .cas, .exch,              // .b32, .b64
           .add,                     // .u32, .s32, .f
           .inc, .dec,               // .u32 only
           .min, .max };             // .u32, .s32, .f
.type  = { .b32, .b64,
           .u32, .u64,
           .s32,
           .f32 };
```

UMA



Dynamic Parallelism



World’s First ARM CPU / CUDA GPU Supercomputer

MONT BLANC

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

http://www.montblanc-project.eu

Mont Blanc Research Project

Exploring energy efficient supercomputer architectures for exascale

ARM CPU + GPU Prototype

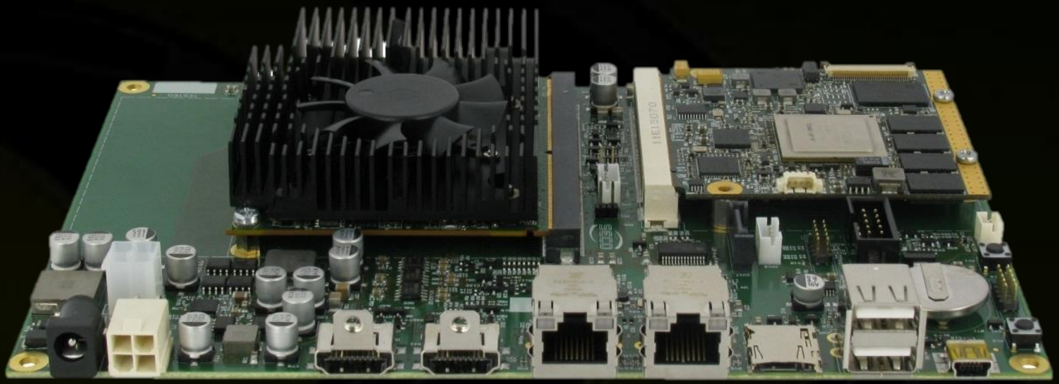
256 Tegra (ARM) CPUs

+ 256 CUDA GPUs

http://www.eesi-project.eu/media/BarcelonaConference/Day2/13-Mont-Blanc_Overview.pdf

CUDA for ARM Development Kit

CUDA GPU Tegra ARM CPU



SECO Hardware Development Kit

http://www.secoqseven.com/en/item/secocq7-mxm/

Research development board

- Quad-core ARM based NVIDIA Tegra 3 processor
- NVIDIA CUDA GPU
- Gigabit Ethernet

CUDA software development kit

Available: 1H 2012