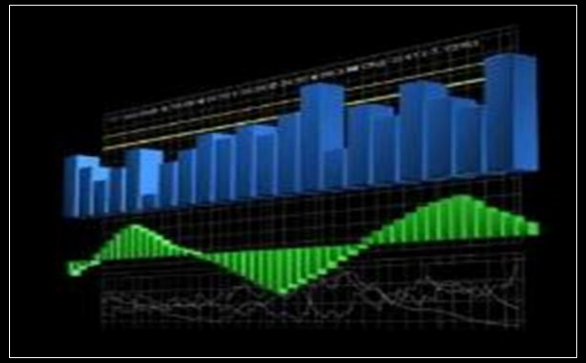
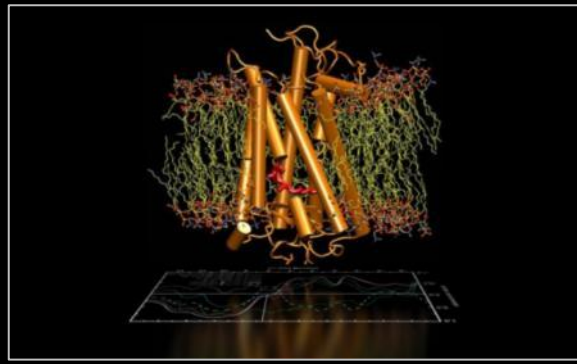
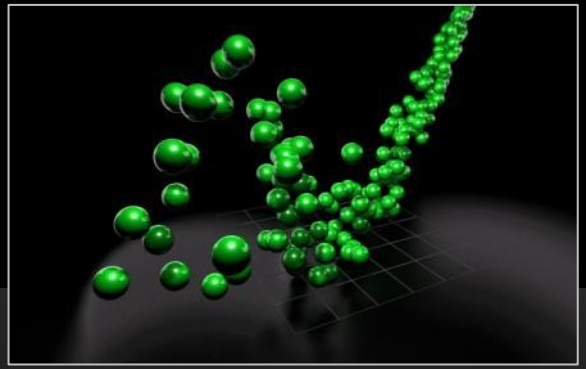
Архитектура NVIDIA Kepler

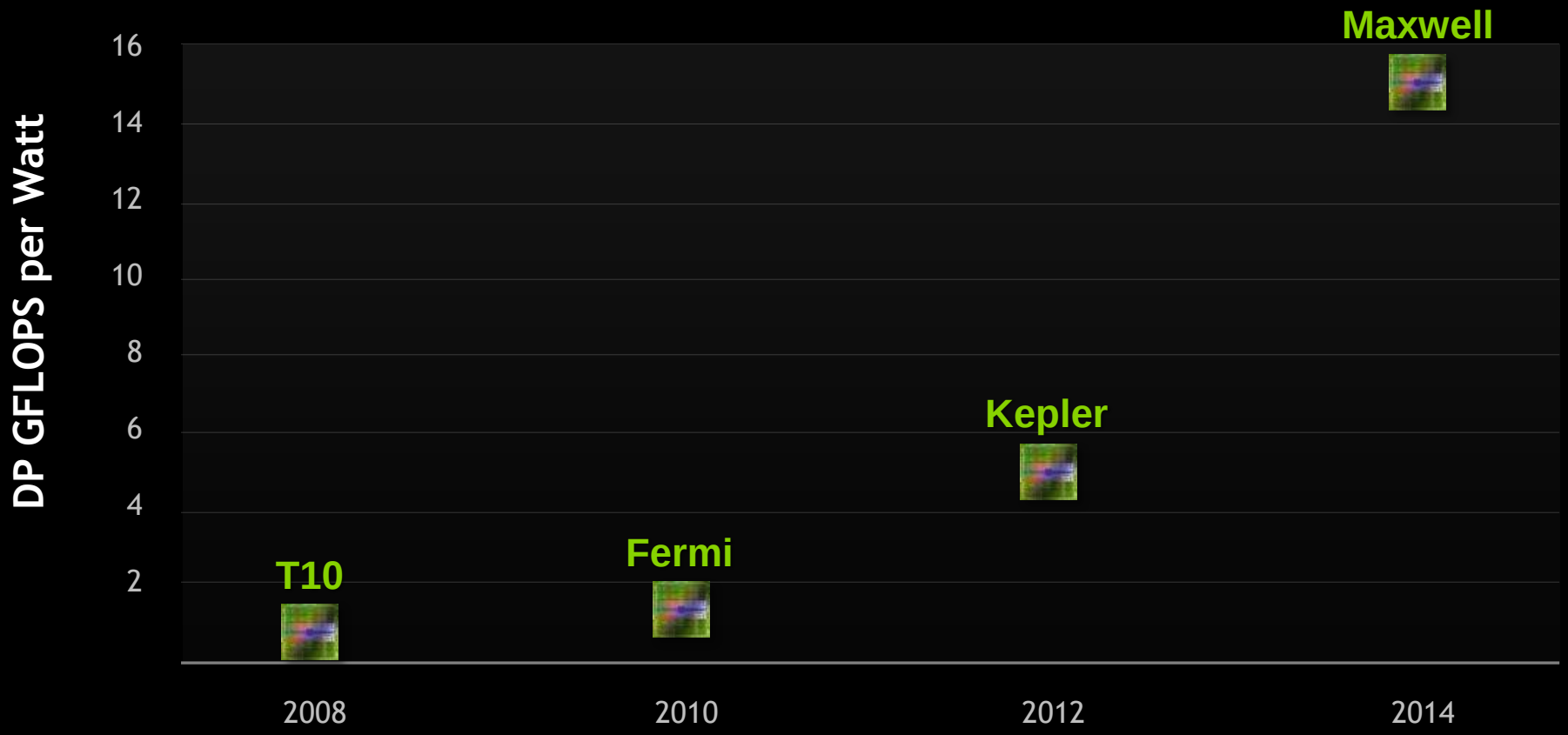






Производительность. Эффективность. Доступность.

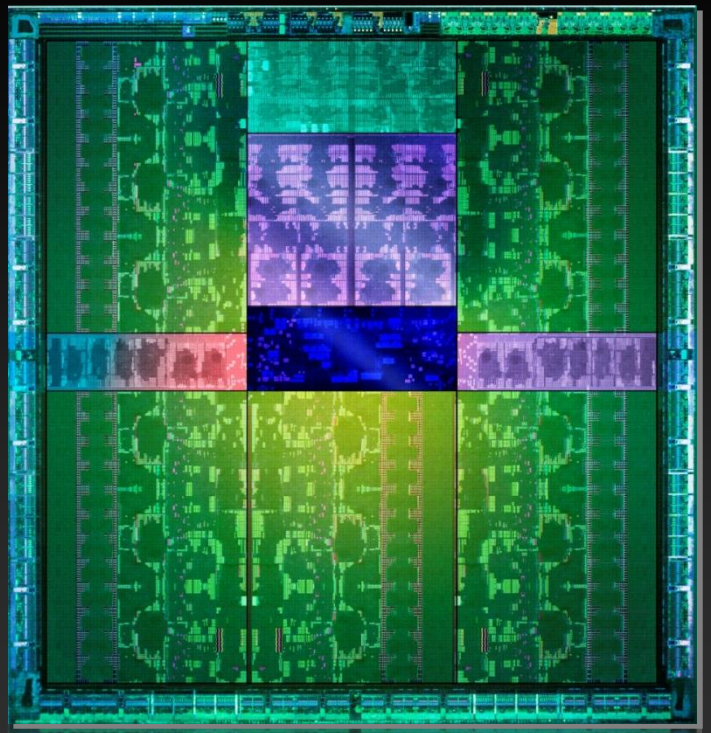
Tesla: в 2-3 раза быстрее каждые 2 года



Kepler

Kepler

САМЫЙ БЫСТРАЯ И ЭФФЕКТИВНАЯ HPC АРХИТЕКТУРА



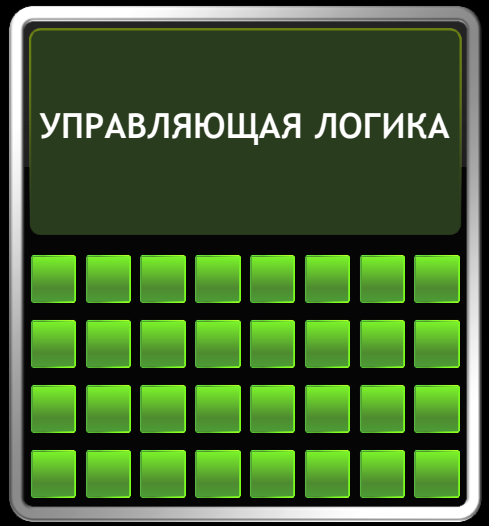
SMX

Hyper-Q

Dynamic Parallelism

Kepler: Скорость и Эффективность

SM
M2090



32 ядра

3x

Perf / Watt

SMX
K20

192 ядра

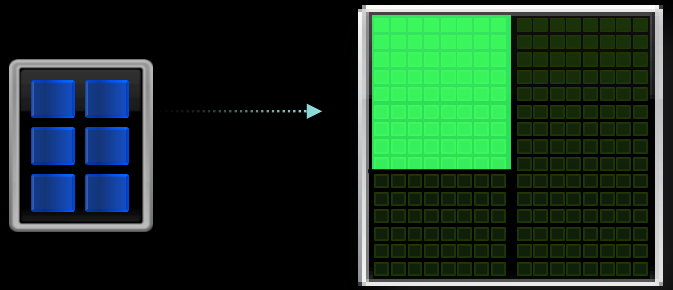
1 Петфлоп
Всего в 10 стойках
400 кВт

Hyper-Q

CPU ядра одновременно запускают задачи на Kepler

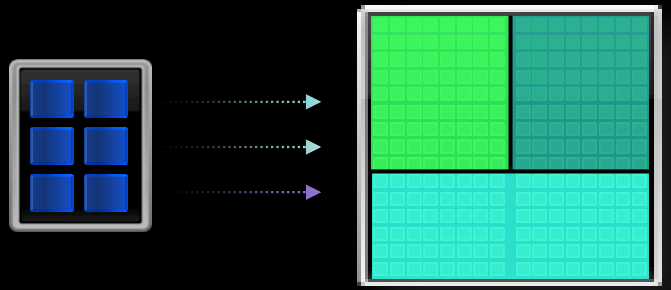
FERMI

1 MPI задача одновременно



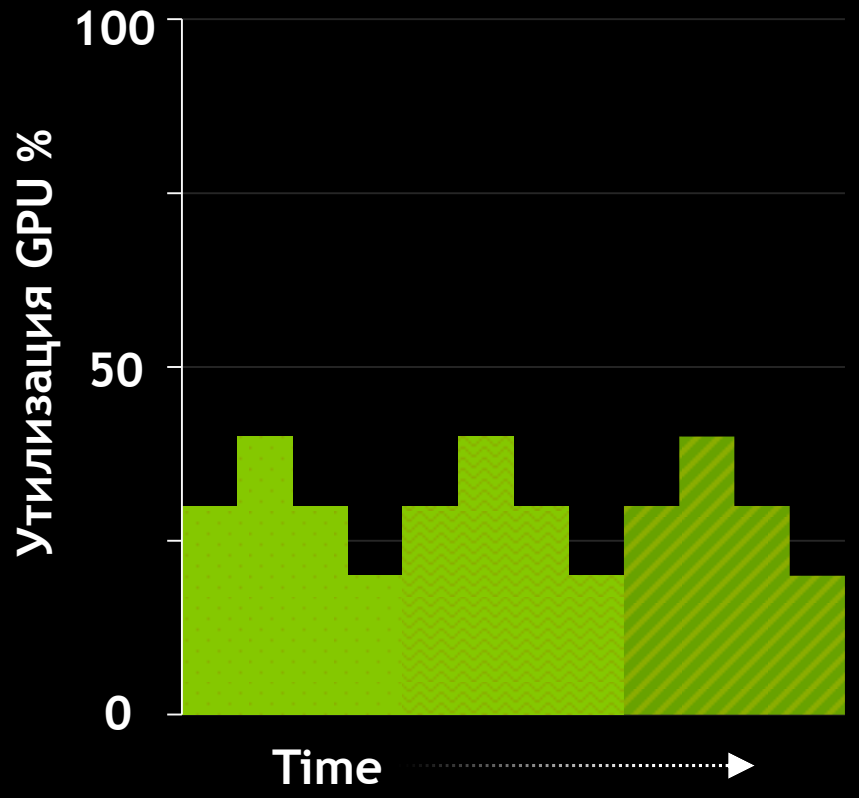
KEPLER

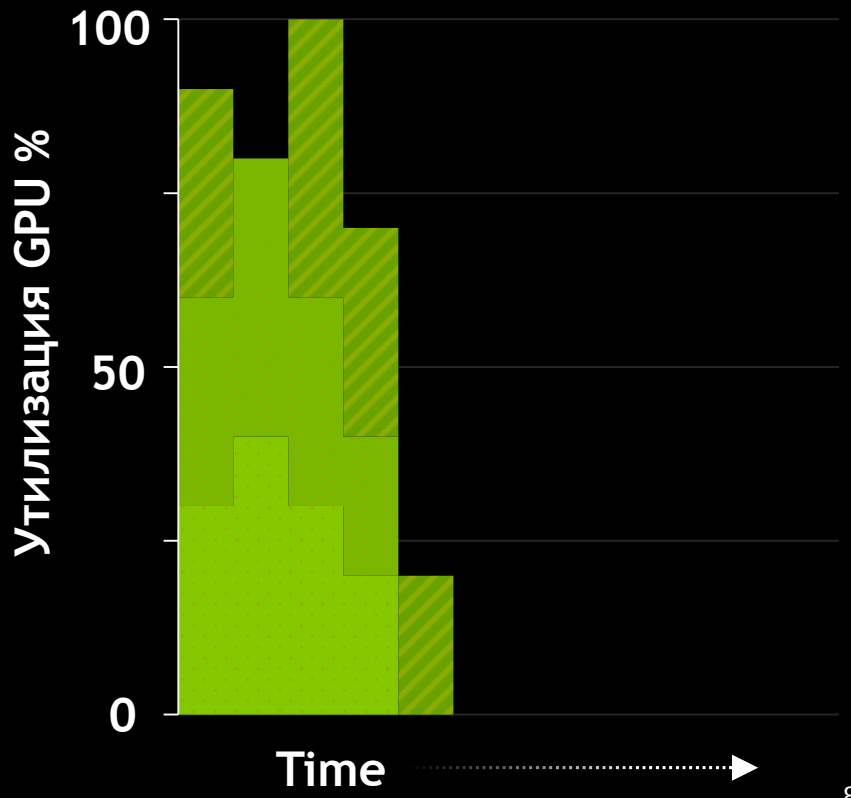
32 MPI задачи одновременно



Hyper-Q

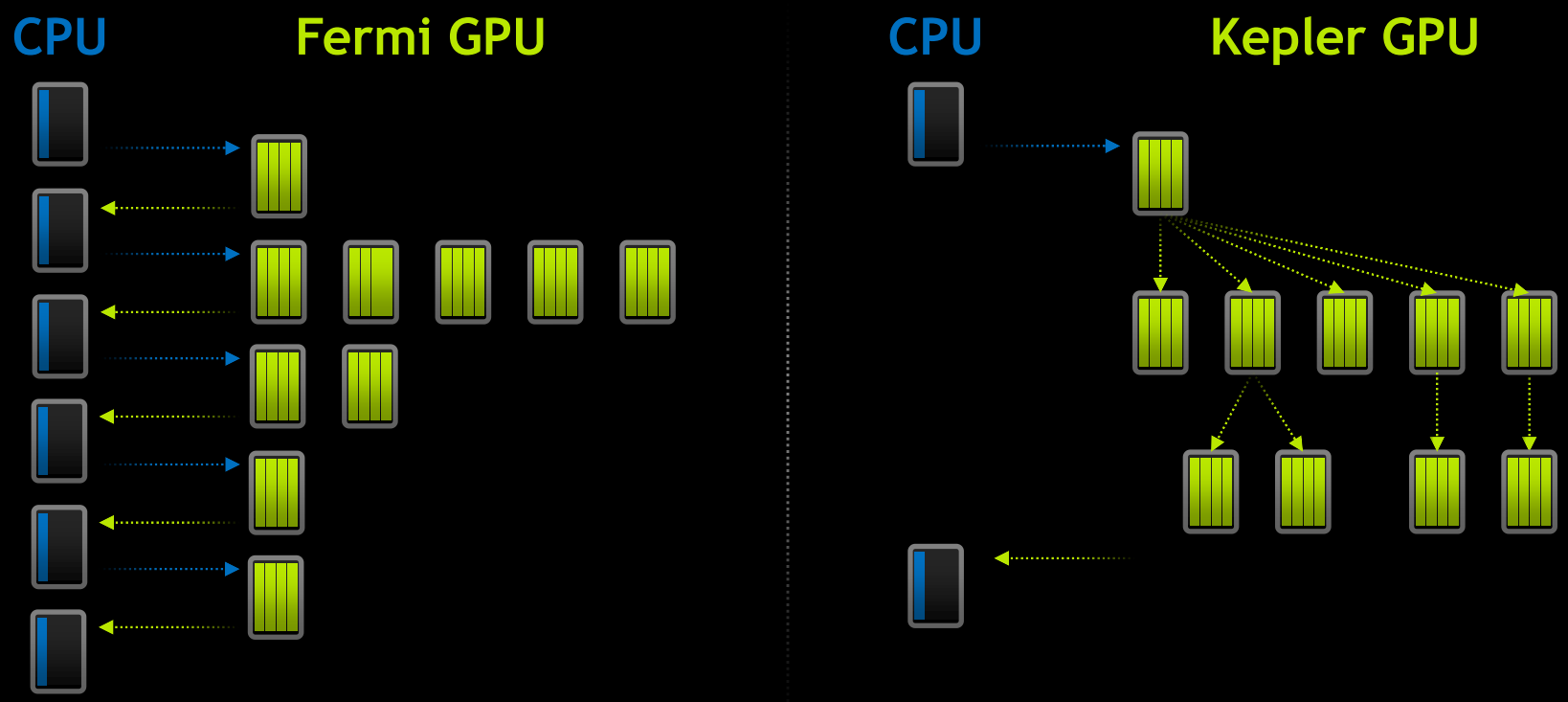
Максимальная утилизация GPU, сокращение времени простоя CPU





Dynamic Parallelism

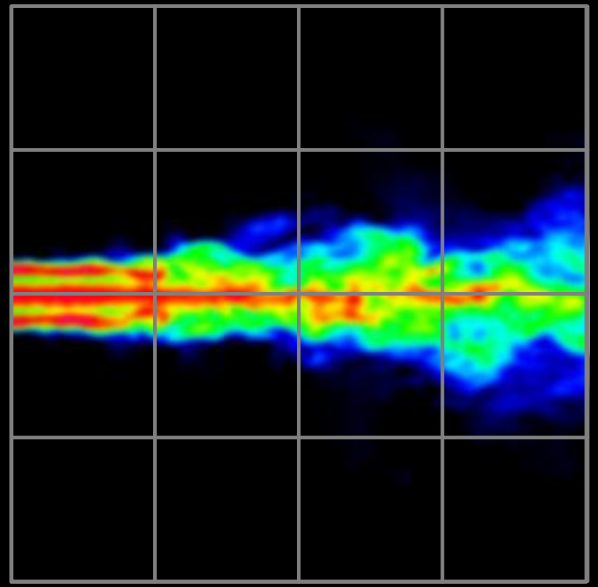
GPU адаптируется к данным, динамически порождая новые потоки



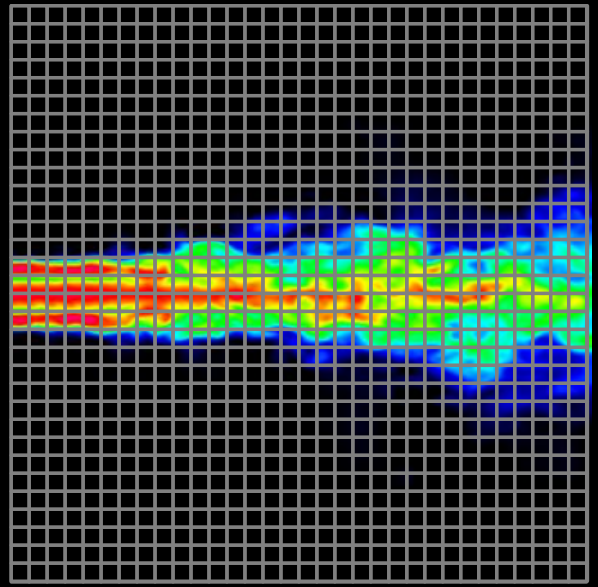
Dynamic Parallelism

Программирование на GPU проще и доступнее

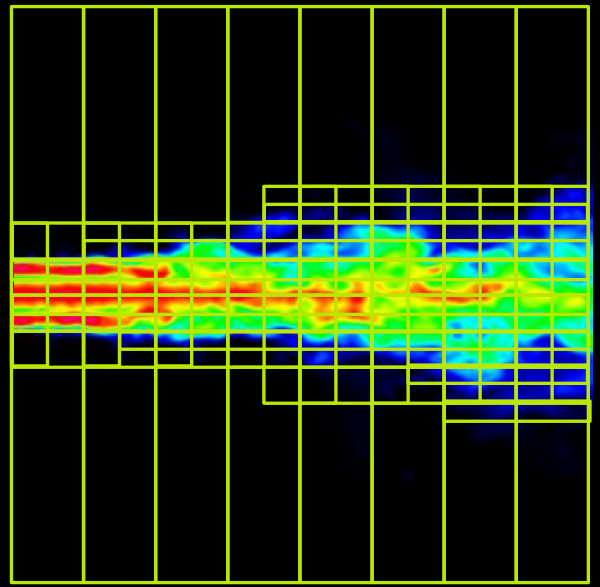
Слишком грубо



Слишком мелко



Как надо



Tesla K10

3x одинарная точность

1.8x пропусная способность памяти

Обработка изображений, сигналов, сейсморазведка

Уже доступно

Tesla K20

3x двойная точность

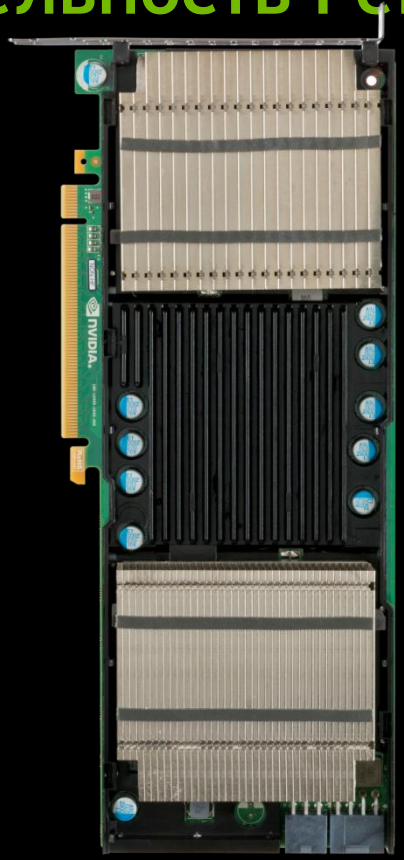
Hyper-Q, Dynamic Parallelism

CFD, FEA, финансы, физика

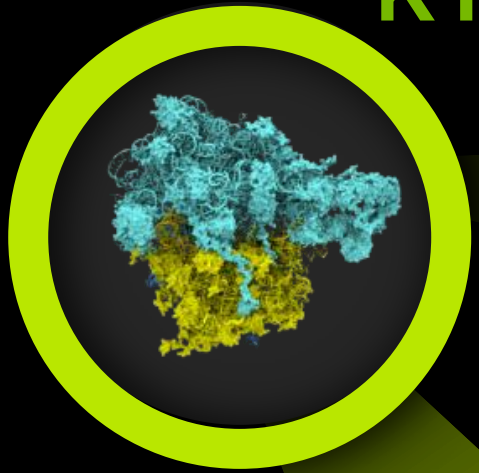
Доступно в Q4 2012

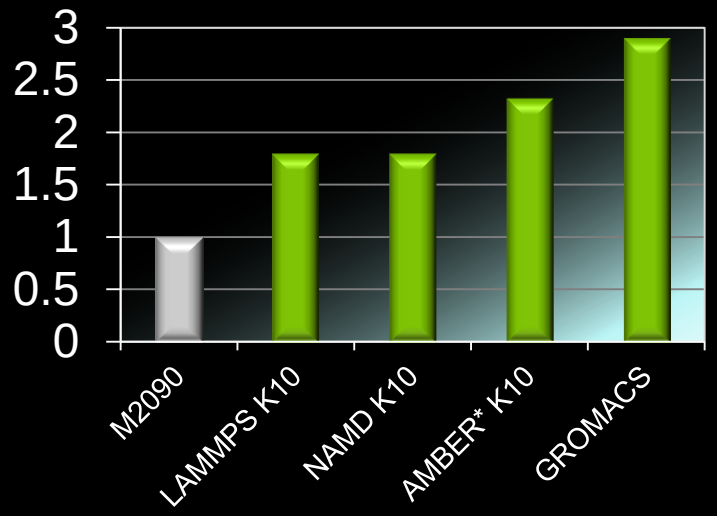
Tesla K10
Тоже потребление, 2x производительность Fermi

Product Name	M2090	K10	
GPU Architecture	Fermi	Kepler GK104	
# of GPUs	1	2	
		Board	Per GPU
Single Precision Flops	1.3 TF	4.58 TF	2.29 TF
Double Precision Flops	0.66 TF	0.190 TF	0.095 TF
# CUDA Cores	512	3072	1536
Memory size	6 GB	8 GB	4GB
Memory BW (ECC off)	177.6 GB/s	320 GB/s	160GB/s
PCI-Express	Gen 2	Gen 3 (Gen 2 compatible)	
Board Power	**225 watts**	**225 watts**	



K10 для биоинформатики

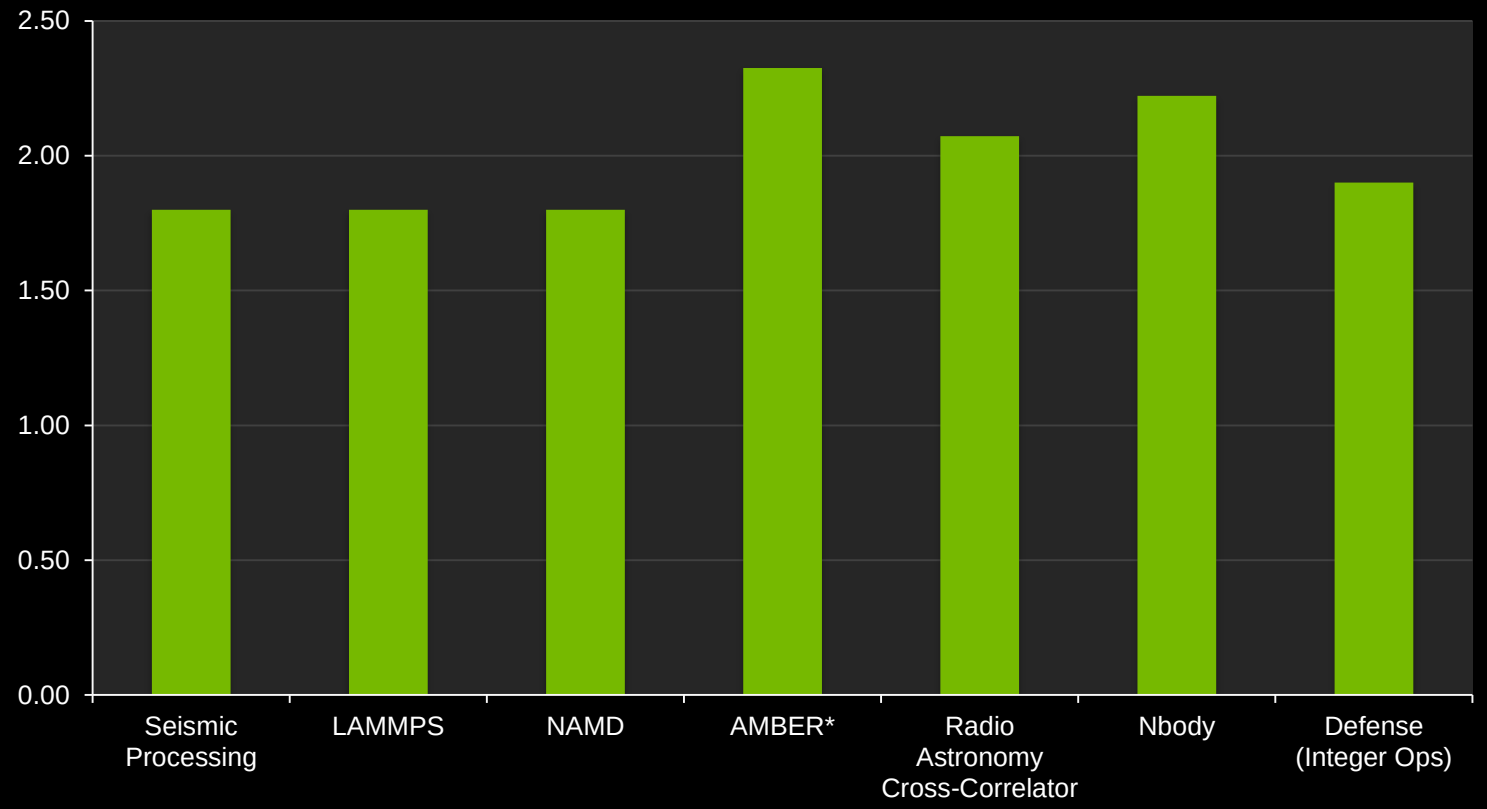




- 2.2X симуляций для приложений МД
- Большие эксперименты на меньших кластерах
- 2X GPU в том же формате

Gromacs 4.6 pre-beta version
* 2 instances of AMBER 12 (with beta patch)

Tesla K10 vs M2090: 2x производительность / Ватт



* 2 instances of AMBER running JAC

Программирование GPU

1 GPU-Accelerated Libraries

Drop in a GPU-accelerated library to replace MKL, IPP, FFTW and other widely-used libraries

Thrust

C++ Template Library

cuBLAS cuSPARSE

Linear Algebra

NPP cuFFT

Signal & Image Processing

More GPU-Accelerated Libaries

2 GPU Directives

Automatically parallelize loops in your Fortran or C code using GPU Directives

OpenACC.
DIRECTIVES FOR ACCELERATORS

- Easy : simply insert hints in your code
- Open : run on either CPU or GPU
- Powerful: tap into the power of GPUs within minutes

Learn More...

Learn More About Directives

3 Programming Languages

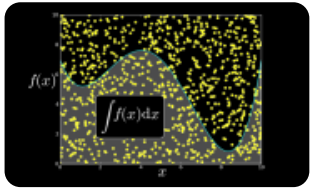
Develop your own parallel applications and libraries using a programming language you already know

CUDA C/C++
GPU Acceleration for
C and C++ Apps.
Learn more...

CUDA Fortran
GPU Acceleration for
Fortran Applications
Learn more...

More Programming Language Solutions

NVIDIA cuBLAS



NVIDIA cuRAND

NVIDIA cuSPARSE

NVIDIA NPP

GPU VSIPL

Vector Signal
Image Processing

CULA tools

GPU Accelerated
Linear Algebra

MAGMA

Matrix Algebra on
GPU and Multicore

NVIDIA cuFFT

ROGUE WAVE SOFTWARE

IMSL Library

CUSP

Sparse Linear
Algebra

libjacket

Building-block
Algorithms for CUDA

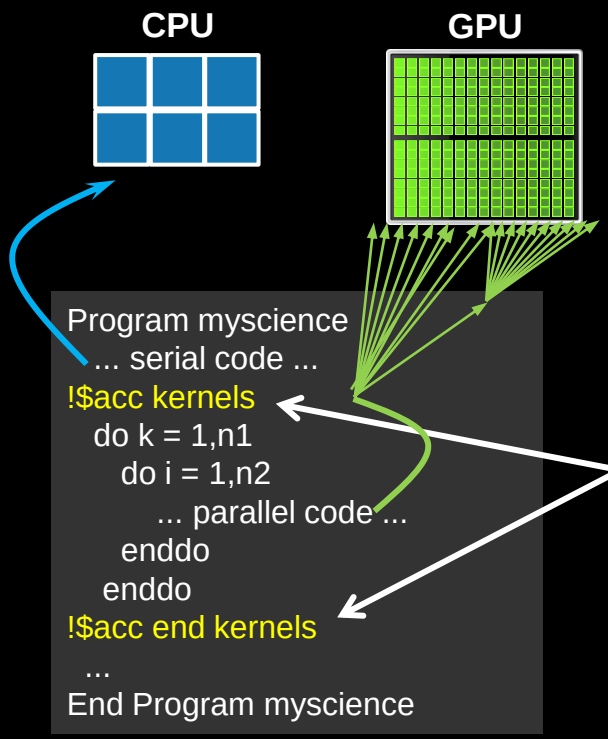
Thrust

C++ STL Features
for CUDA

Библиотеки для GPU

“Copy-paste” для ускорения приложений

Директивы OpenACC



Исходный код на C/Fortran

OpenACC метки для компилятора

Простые указатели для компилятора

Компилятор параллелизует код

Работает на многоядерных CPU и массивно параллельных GPU

Минимум усилий. Ощутимый результат

Модель жизненного цикла морской фауны

Университет Мельбурна

65x за 2 для

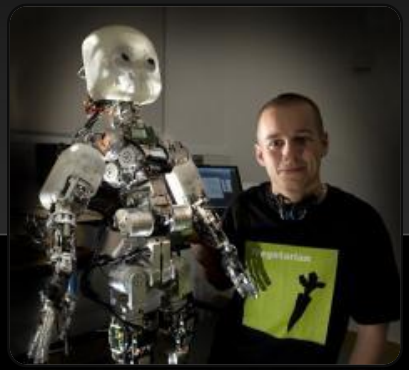
Звезды и галактики 12.5 млрд лет назад

Университет Гронингена

5.6x за 5 дней

Нейросети для самообучаемых роботов

Университет Плимута



4.7x за 4 часа

Воркшоп по OpenACC в суперкомпьютерном центре Питсбурга

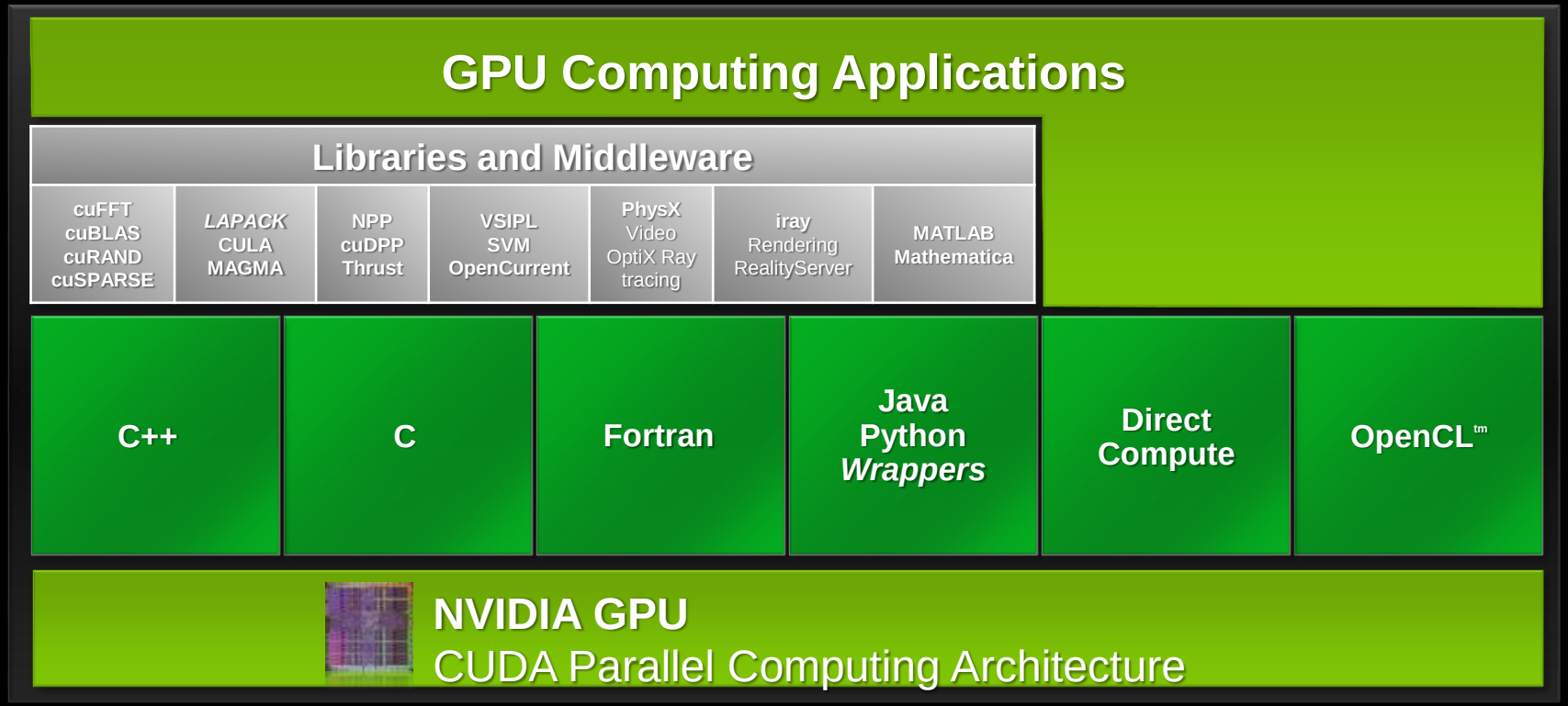
К концу второго дня

получено 10-кратное ускорение одного из атмосферных ядер

6 директив

Technology Director
National Center for Atmospheric Research (NCAR)

Поддержка языков C, C++, Fortran моделью параллельного программирования CUDA



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

C для CUDA : C + «синтаксический сахар»

```
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);
```

Стандартный код C

```
__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 nblocks = (n + 255) / 256;
saxpy_parallel<<<nblocks, 256>>>(n, 2.0, x, y);
```

Параллельный код C

Приложения

Название	GPU функционал
AMBER 11	Simulation of mechanics force fields
NAMD	High-performance simulation of large molecular systems
GROMACS	Proteins and lipids with bonded interactions
Schrodinger	Core Hopping, PRIME (2011)
OpenEye	fastROCS
CUDA-BLASTP	Scanning Protein Sequence Database
CUDASW++	Smith-Waterman Protein Sequence Database
HMMER	HMMER Protein Sequence Analysis
VMD	Molecular Visualization in 3D
CHARMM	Molecular Modeling (2011)
GAMESS	Quantum Chemistry (2011)