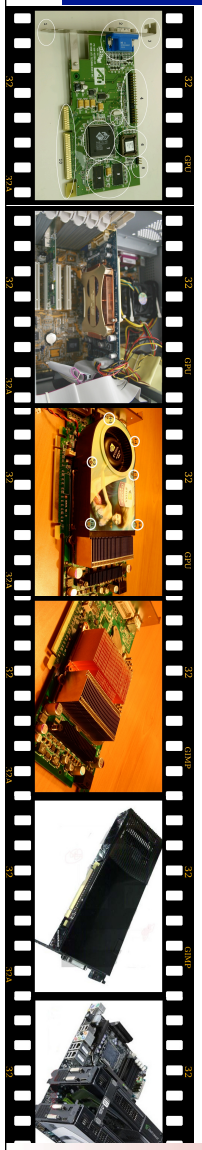
HPC Advisory Council Spain Conference 2012

Programming models for heterogeneous computing

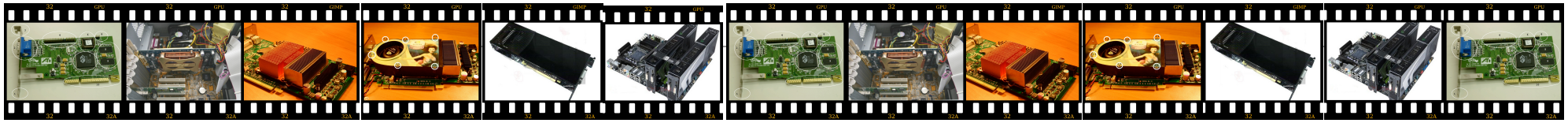
Manuel Ujaldón
Nvidia CUDA Fellow and A/Prof.
Computer Architecture Department
University of Malaga

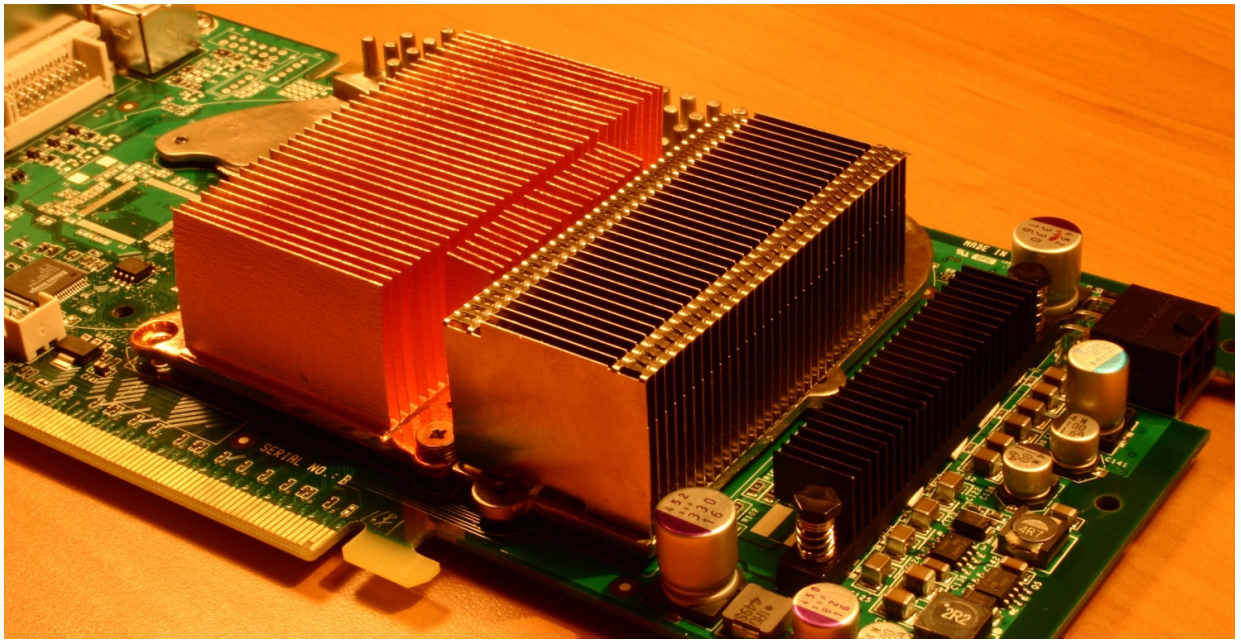
Talk outline [30 slides]



1. Introduction [5 slides]
2. The GPU evolution [5 slides]
3. Programming [11 slides]
    1. Libraries [2 slides]
    2. Switching among hardware platforms [4 slides]
    3. Accessing CUDA from other languages [1 slide]
    4. OpenACC [4 slides]
4. The new hardware [9 slides]
    1. Kepler [8 slides]
    2. Echelon [1 slide]

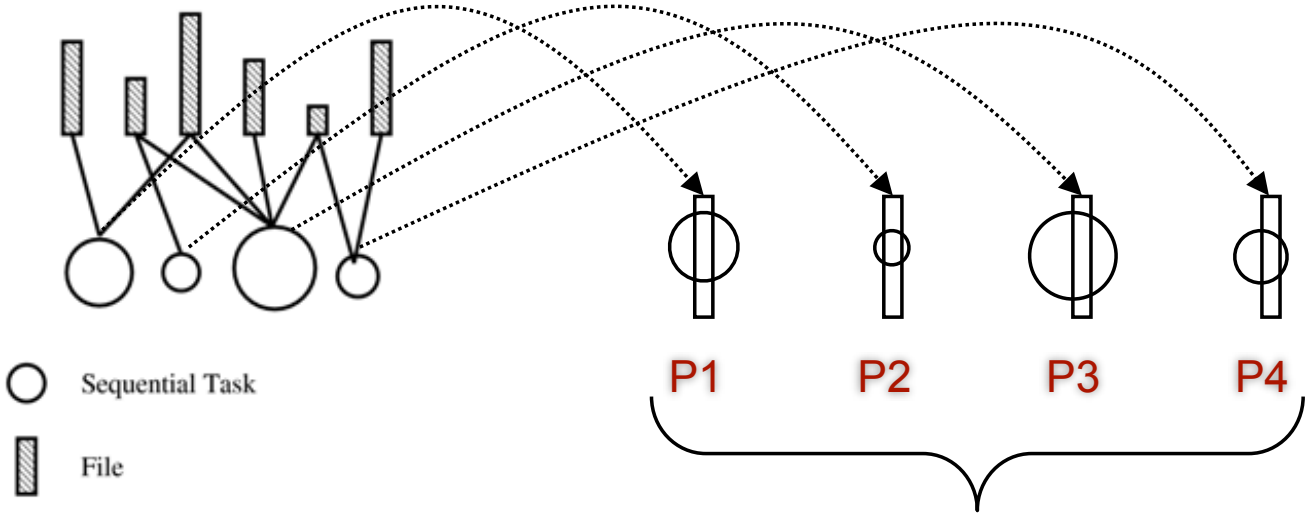
I. Introduction





An application which favours CPUs: Task parallelism and/or intensive I/O

When applications are bags of tasks (few): Apply task parallelism

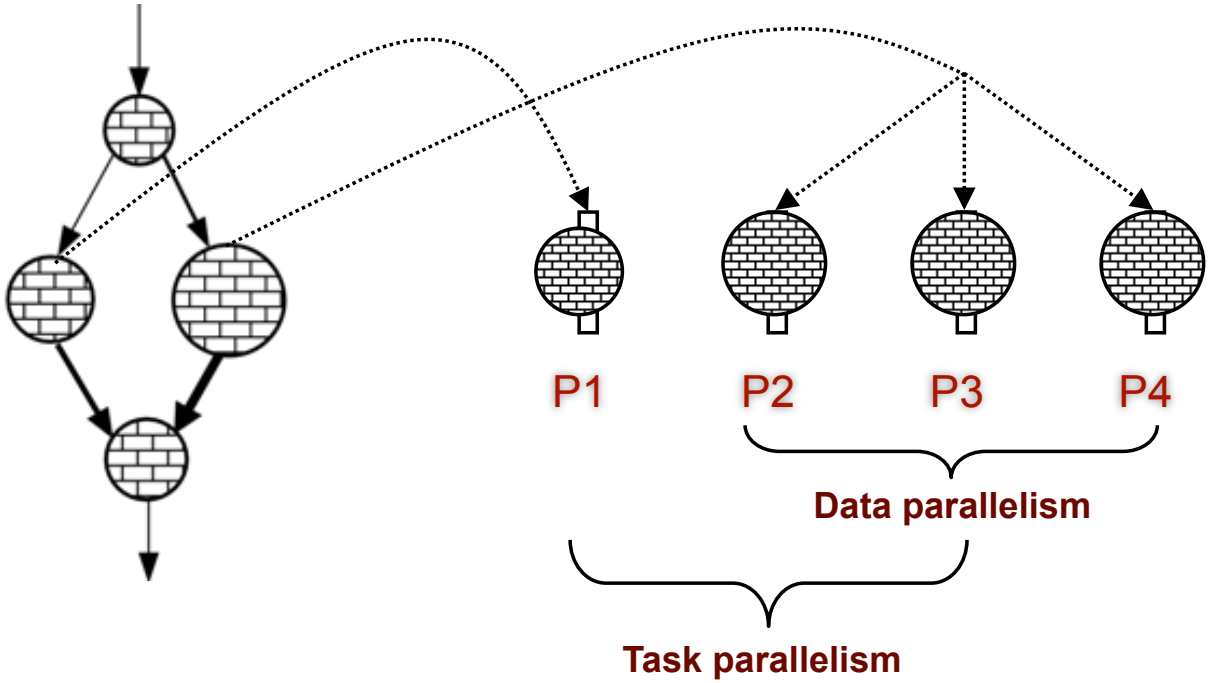


Try to balance tasks while keeping relation to disk files

An application which favours GPUs: Data parallelism [+ large scale]

Sequential or Parallel Task

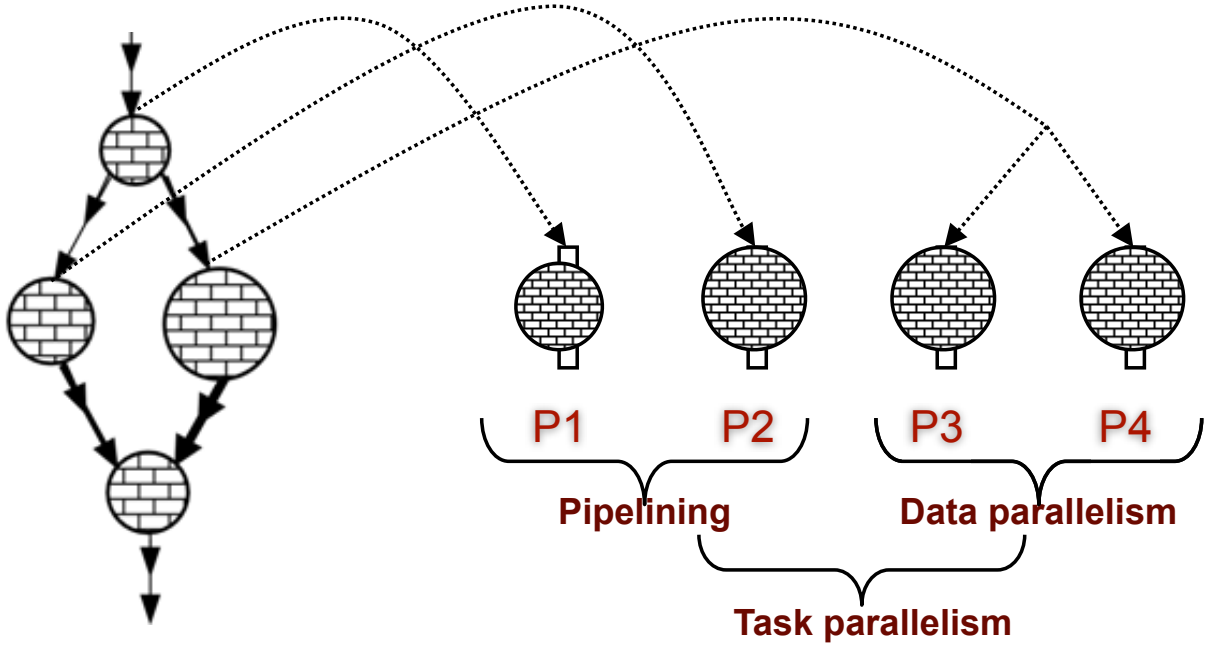
When applications are no streaming workflows: Combine task and data parallelism



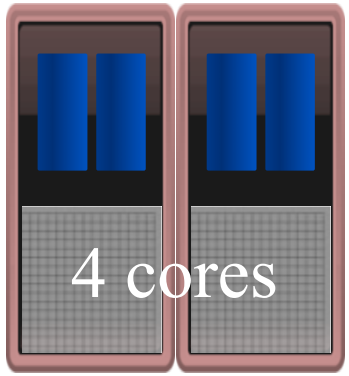
Heterogeneous case: More likely, requires a wise programmer to exploit each processor

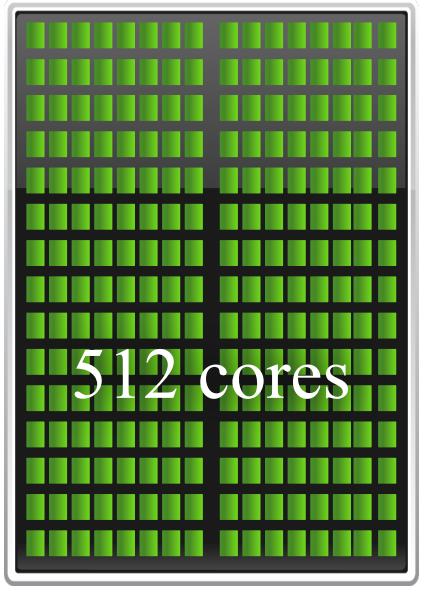
Sequential or Parallel Task

When applications are streaming workflows: Task parallelism, data parallelism, and pipelining

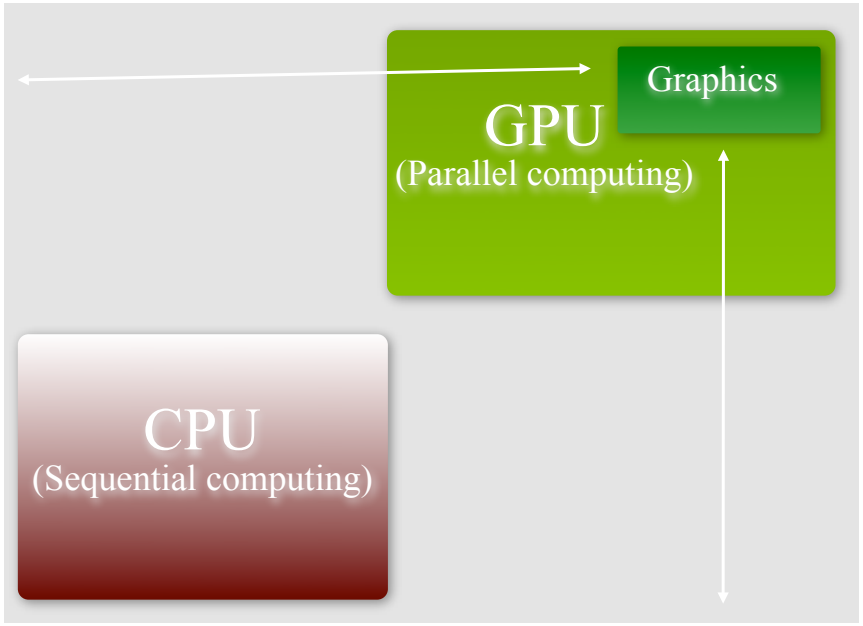


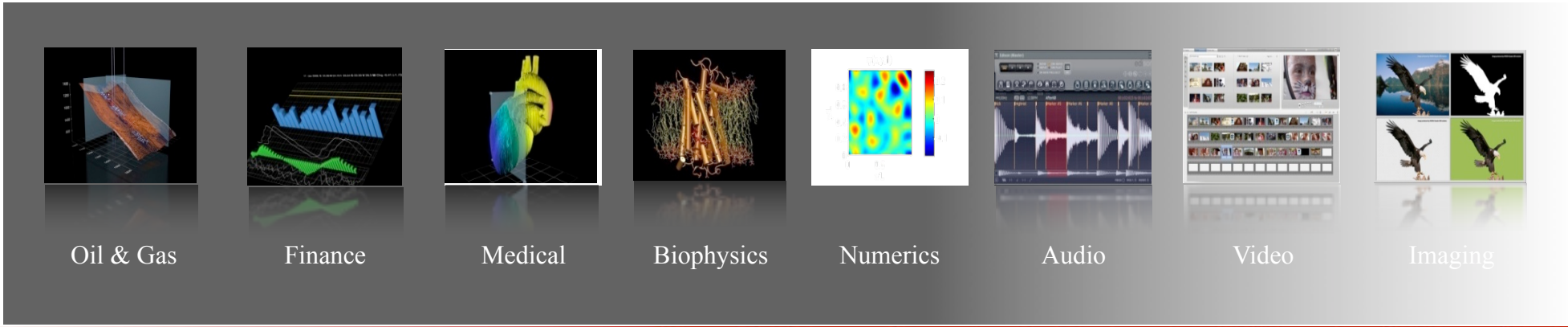
Hardware resources and scope of application for the heterogeneous model



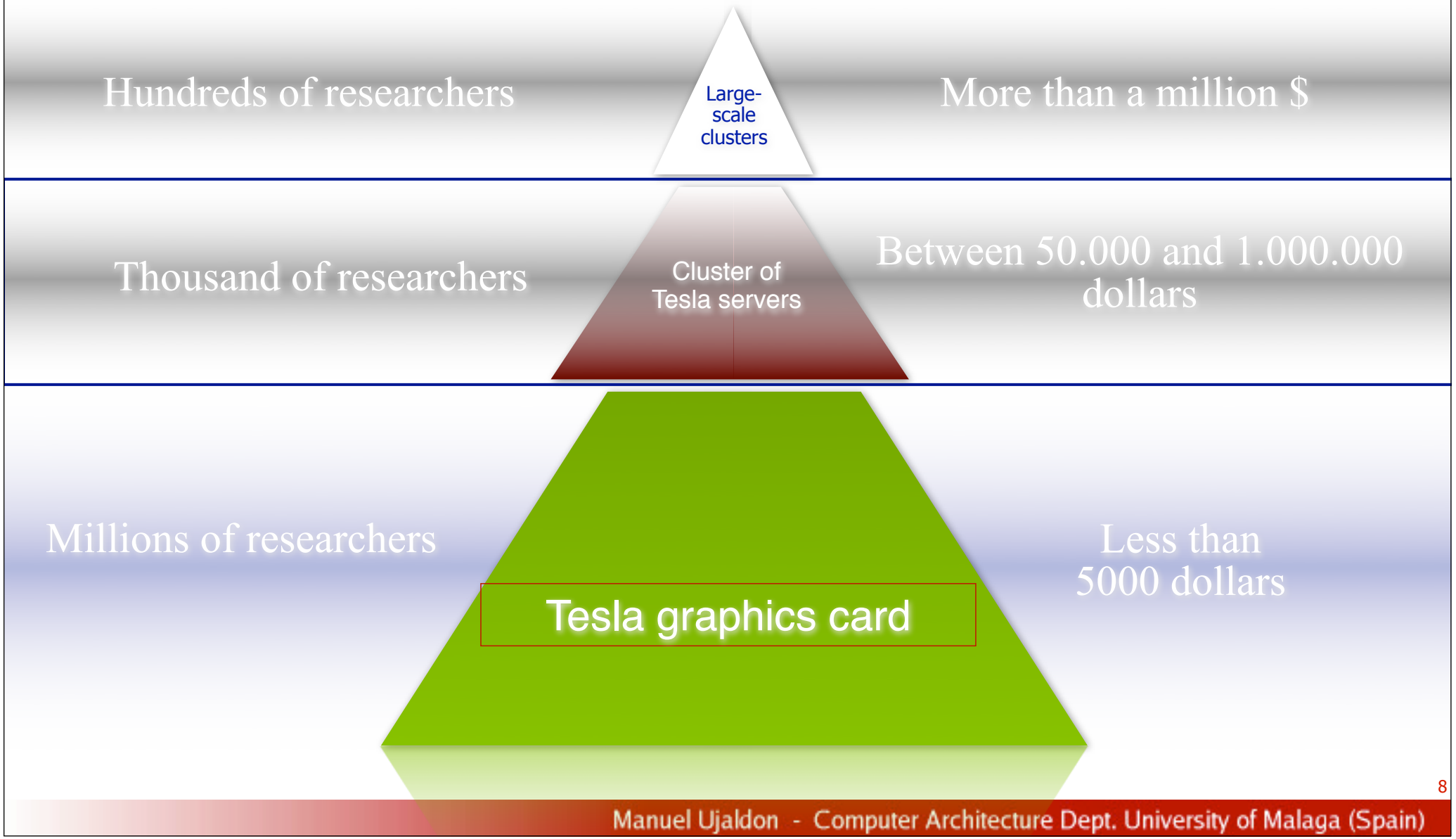


Use CPU and GPU:
Every processor executes those parts where it gets more effective

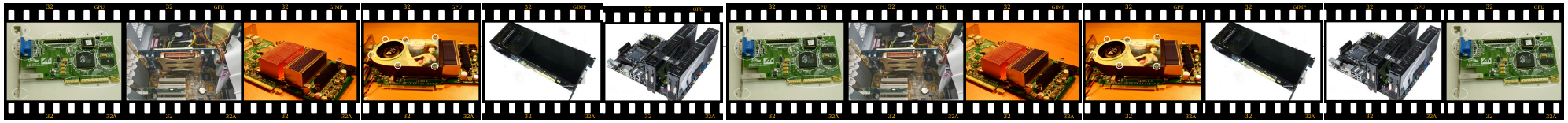


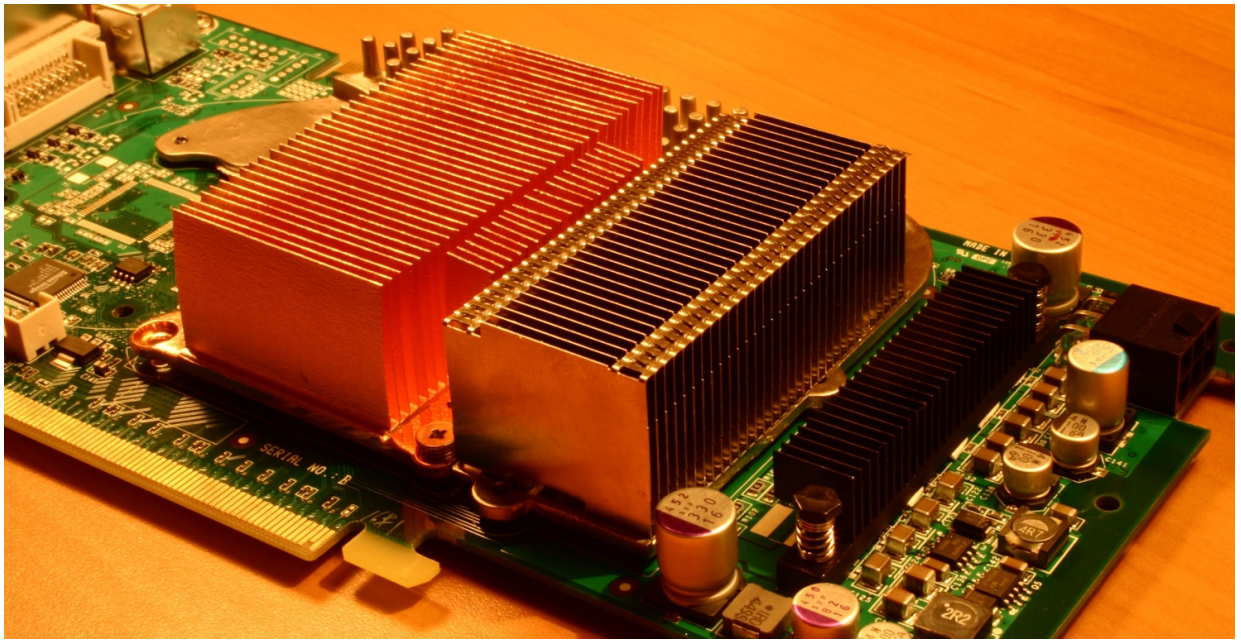


There is a hardware platform for each end user



II. The GPU evolution



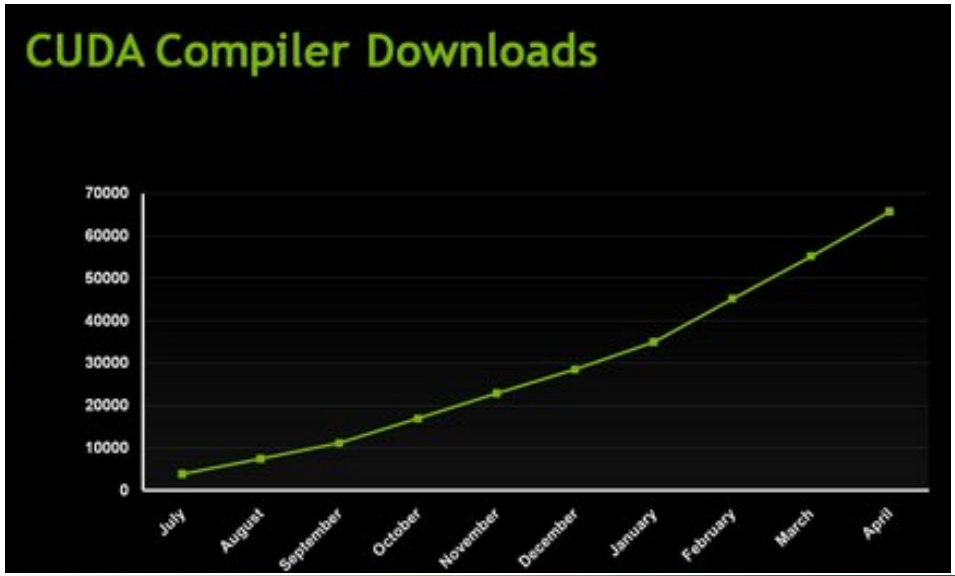


The graphics card within the domestic hardware marketplace (regular PCs)

- GPUs sold per quarter:
  - 114 millions [Q4 2010]
  - 138.5 millions [Q3 2011]
  - 124 millions [Q4 2011]
- The marketplace keeps growing, despite of global crisis.
- Compared to CPUs sold, 93.5 millions [Q4 2011], there are 1.5 GPUs out there for each CPU, and this factor keeps growing relentlessly over the last decade (it was barely 1.15x in 2001).

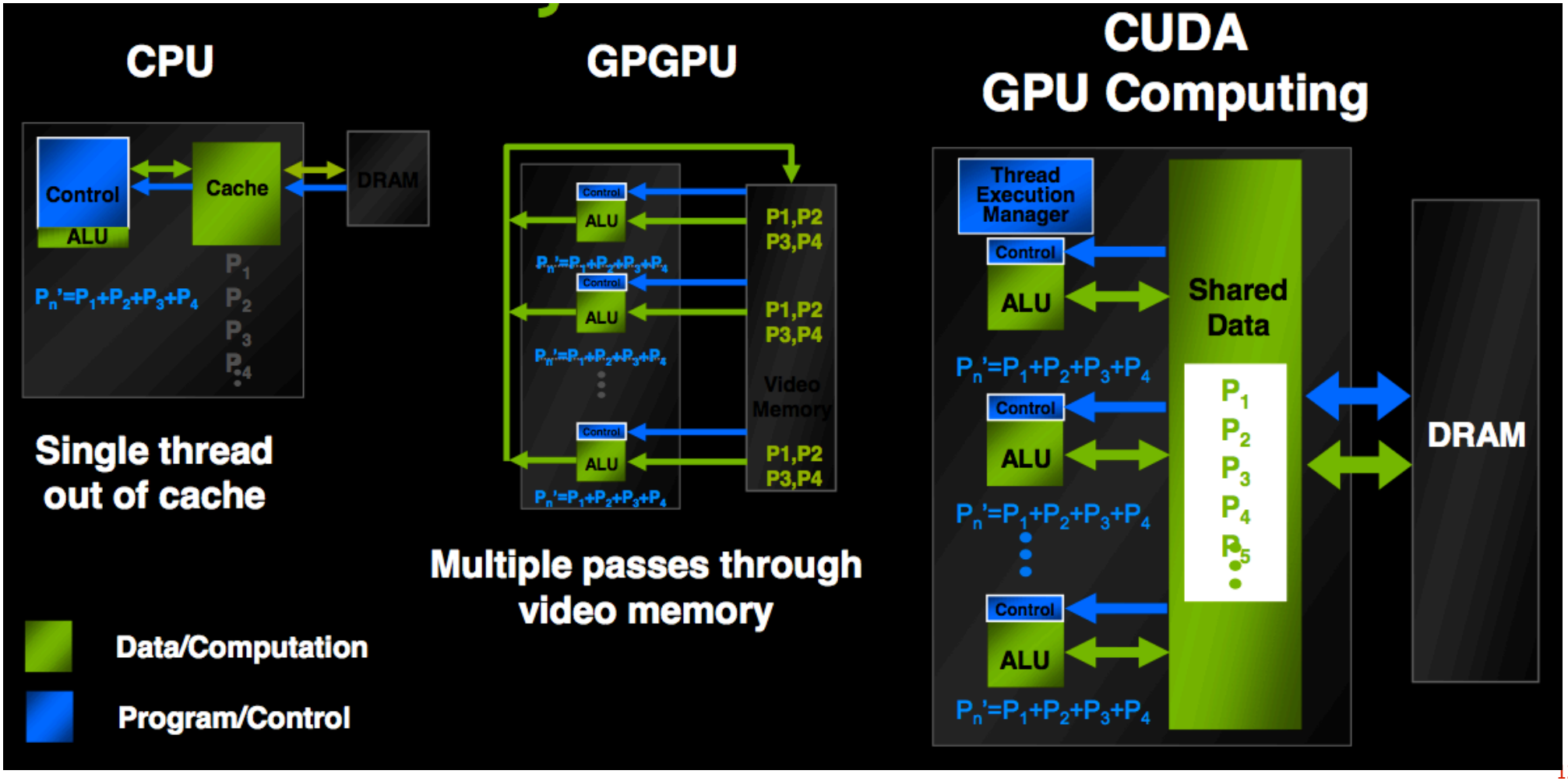
In barely 5 years, CUDA programming has grown to become ubiquitous

- More than 500 research papers are published each year.
- More than 500 universities teach CUDA programming.
- More than 350 million GPUS are programmed with CUDA.
- More than 150.000 active programmers.
- More than a million compiler and toolkit downloads.

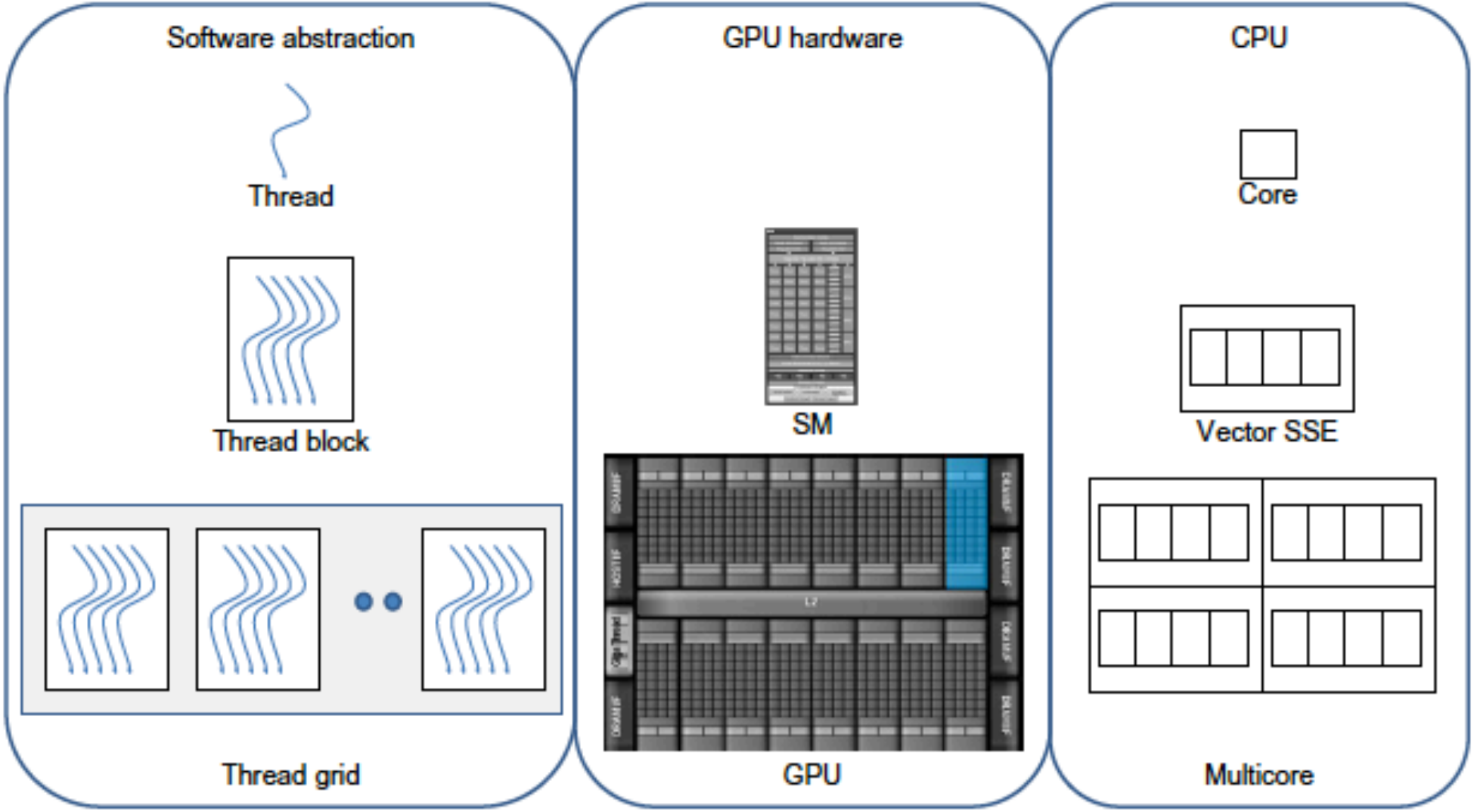


The three generations of processor design

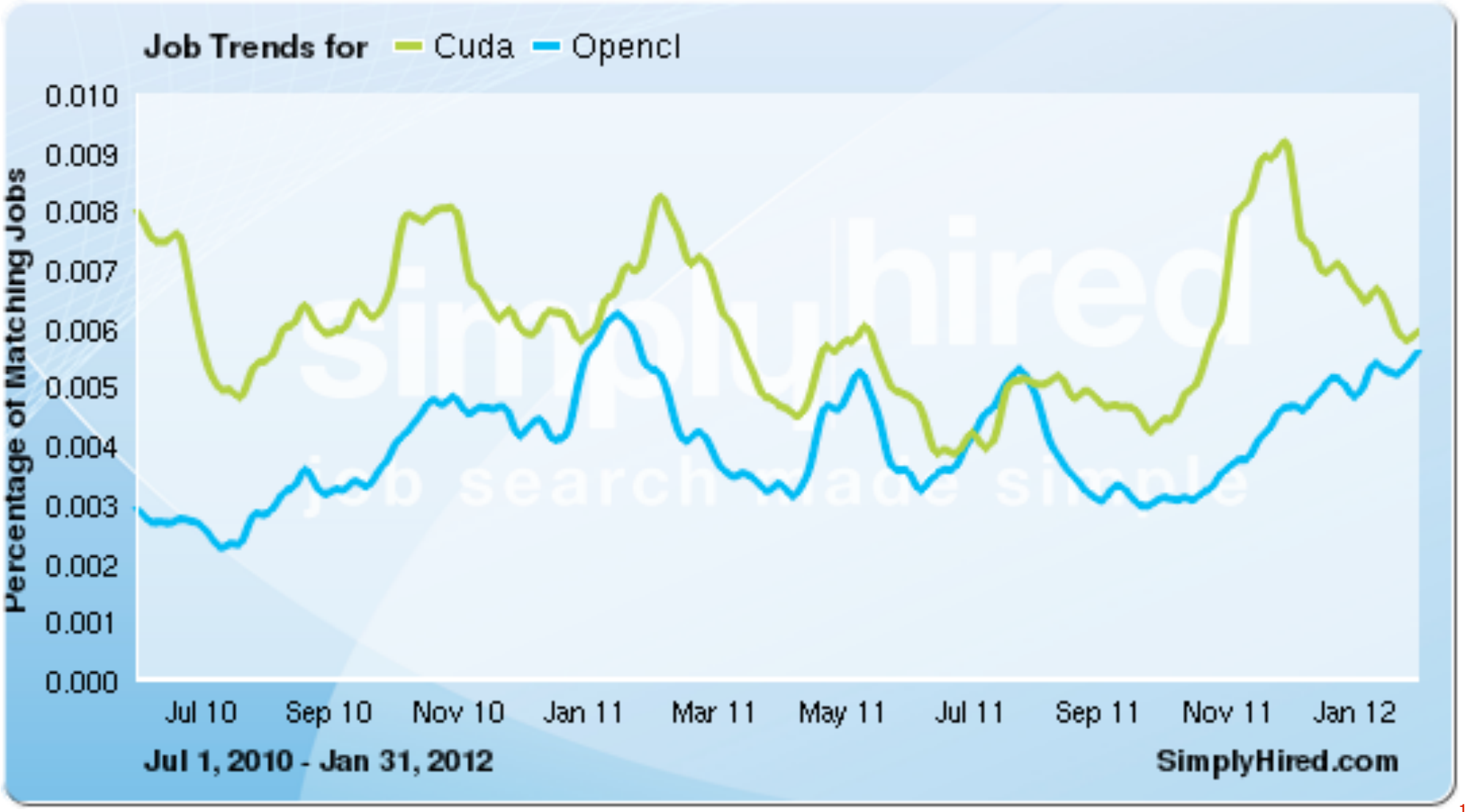
Before 2005 | 2005 - 2007 | 2008 - 2012



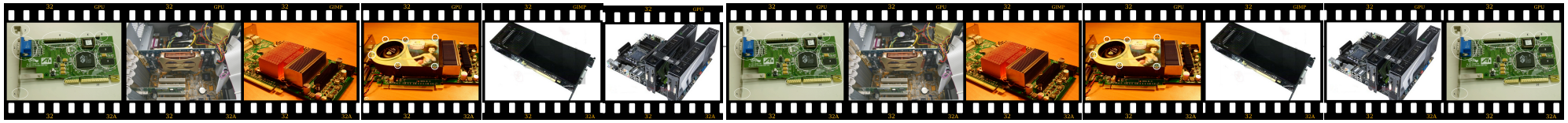
… and how they are connected to programming trends

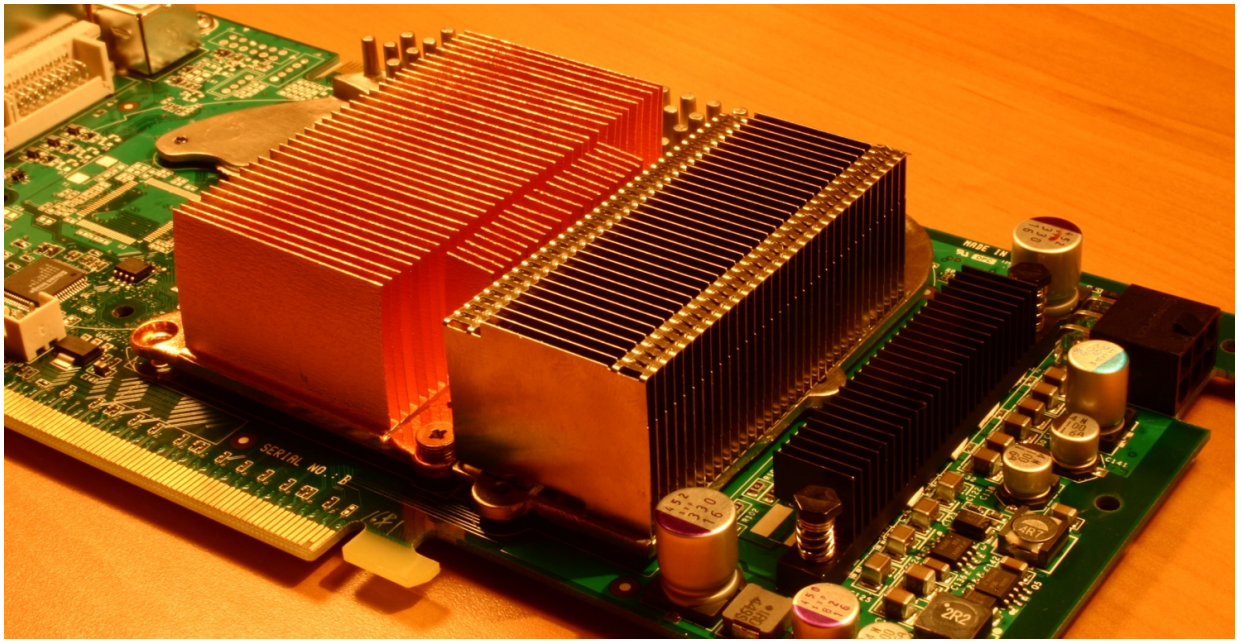


We also have OpenCL, which extends GPU programming to non-Nvidia platforms



III. Programming





III. 1. Libraries

A brief example. Google search is a must before starting an implementation

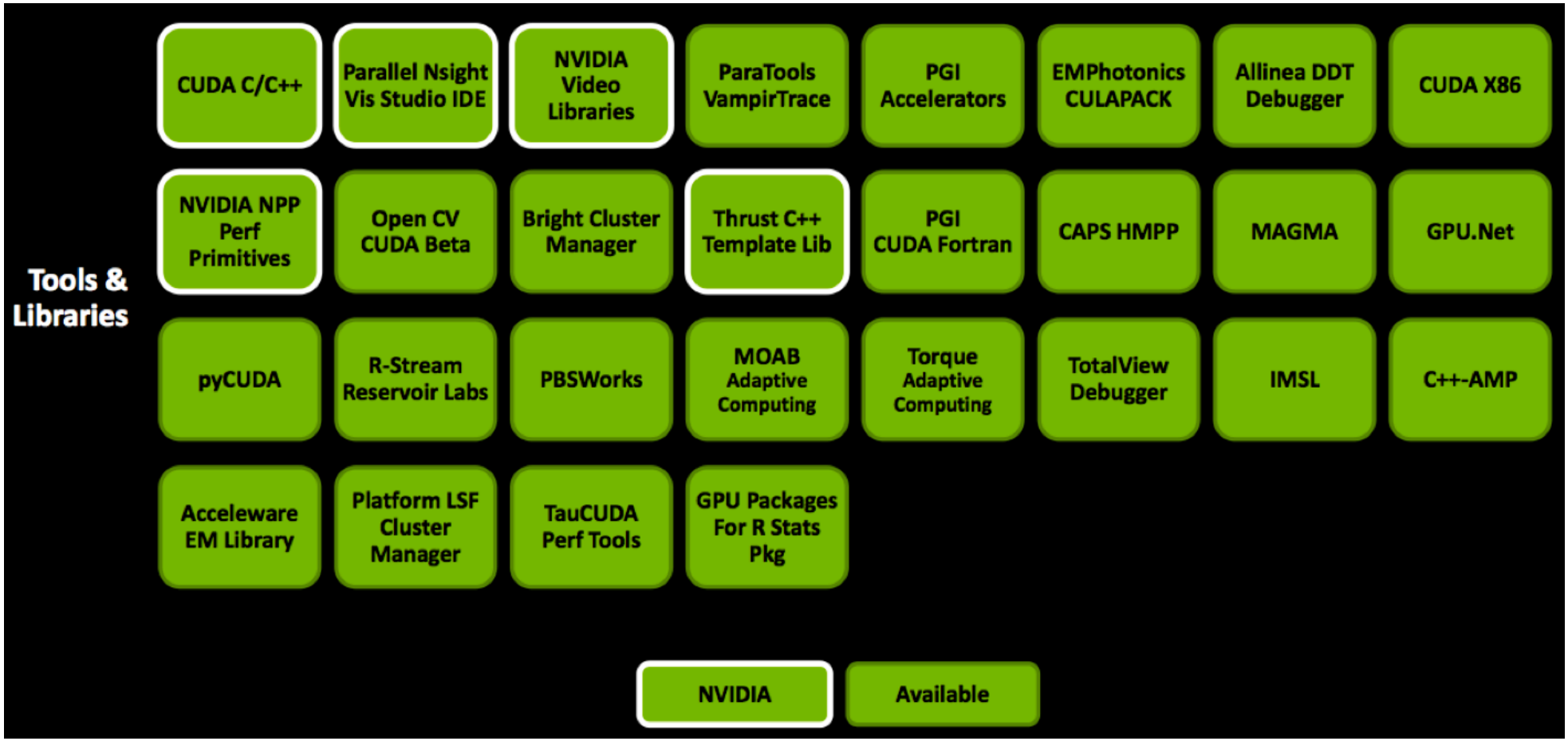
CUDA Math Libraries

High performance math routines for your applications:

- cuFFT - Fast Fourier Transforms Library
- cuBLAS - Complete BLAS Library
- cuSPARSE - Sparse Matrix Library
- cuRAND - Random Number Generation (RNG) Library
- NPP - Performance Primitives for Image & Video Processing
- Thrust - Templated Parallel Algorithms & Data Structures
- math.h - C99 floating-point Library

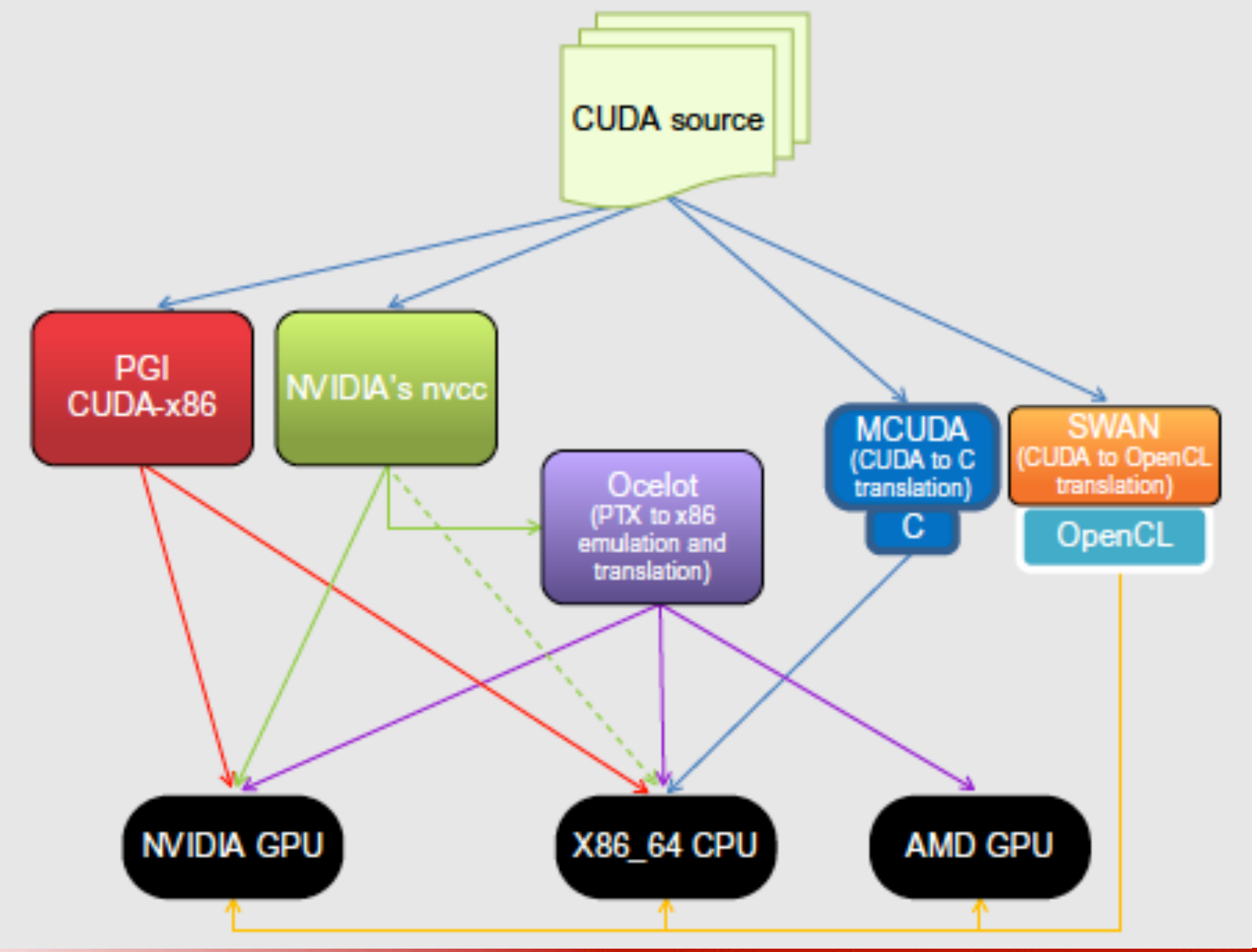
Included in the CUDA Toolkit Free download @ www.nvidia.com/getcuda

The developer ecosystem enables the application growth



III. 2. Switching among hardware platforms

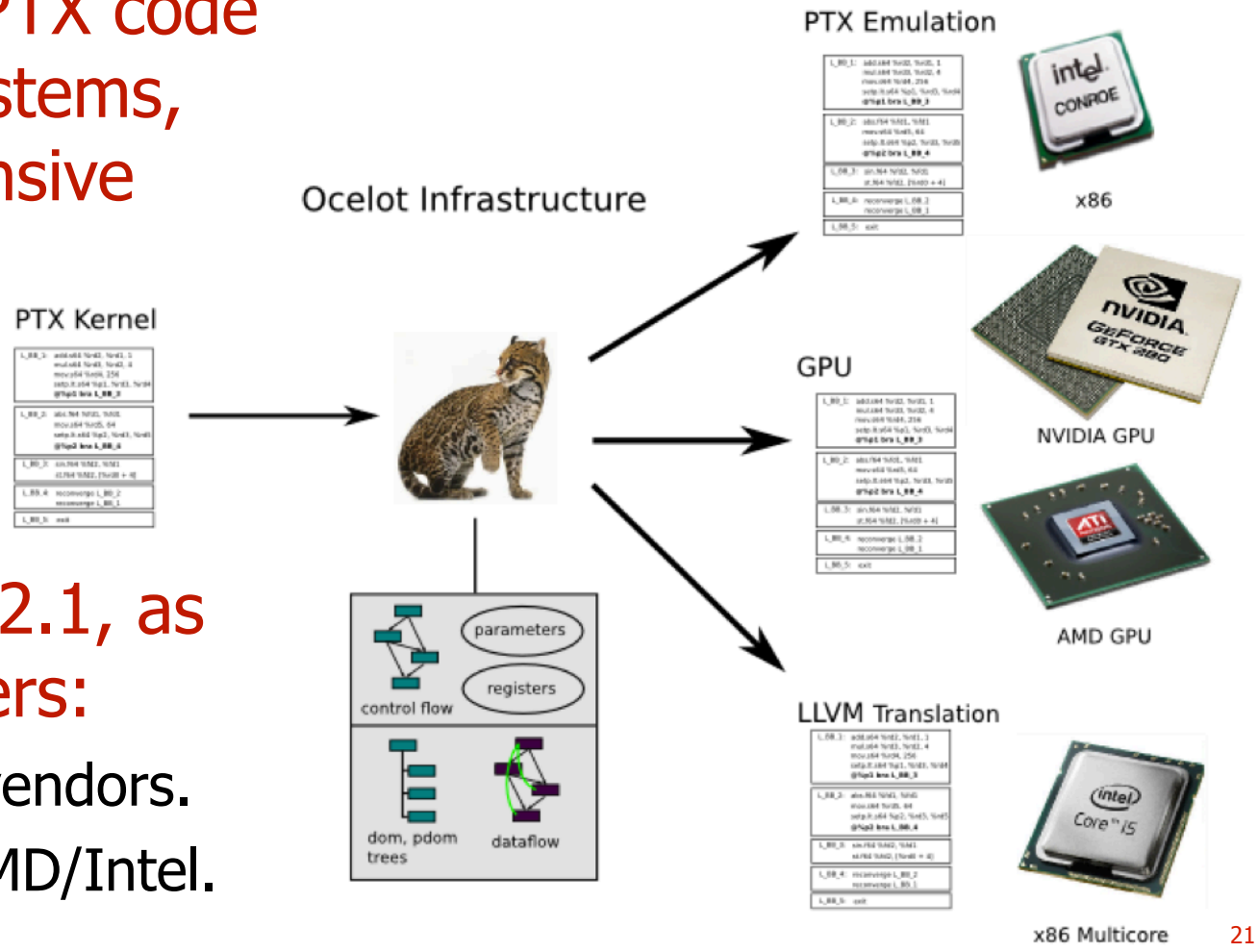
Compiling for other target platforms



Ocelot

http://code.google.com/p/gpuocelot

- It is a dynamic compilation environment for the PTX code on heterogeneous systems, which allows an extensive analysis of the PTX code and its migration to other platforms.
- The latest version (2.1, as of April 2012) considers:
  - GPUs from multiple vendors.
  - x86-64 CPUs from AMD/Intel.



Swan
http://www.multiscalelab.org/swan

- A source-to-source translator from CUDA to OpenCL:
  - Provides a common API which abstracts the runtime support of CUDA and OpenCL.
  - Preserves the convenience of launching CUDA kernels (<<<grid,block>>>), generating source C code for the entry point kernel functions.
  - ... but the conversion process is not automatic and requires human intervention.
- Useful for:
  - Evaluate OpenCL performance for an already existing CUDA code.
  - Reduce the dependency from `nvcc` when we compile host code.
  - Support multiple CUDA compute capabilities on a single binary.
  - As runtime library to manage OpenCL kernels on new developments.

PGI CUDA x86 compiler http://www.pgroup.com

- Major differences with previous tools:
  - It is not a translator from the source code, it works at runtime. In 2012, it will allow to build a unified binary which will simplify the software distribution.
- Main advantages:
  - Speed: The compiled code can run on a x86 platform even without a GPU. This enables the compiler to vectorize code for SSE instructions (128 bits) or the most recent AVX (256 bits).
  - Transparency: Even those applications which use GPU native resources like texture units will have an identical behavior on CPU and GPU.

III. 3. Accessing CUDA from other languages

Some possibilities

- CUDA can be incorporated into any language that provides a mechanish for calling C/C++. To simplify the process, we can use general-purpose interface generators.
- SWIG [http://swig.org] (Simplified Wrapper and Interface Generator) is the most renowned approach in this respect. Actively supported, widely used and already successful with: AllegroCL, C#, CFFI, CHICKEN, CLISP, D, Go language, Guile, Java, Lua, MxScheme/Racket, Ocaml, Octave, Perl, PHP, Python, R, Ruby, Tcl/Tk.
- A connection with Matlab interface is also available:
  - On a single GPU: Use Jacket, a numerical computing platform.
  - On multiple GPUs: Use MatWorks Parallel Computing Toolbox.

III. 4. OpenACC

The OpenACC initiative

OpenACC: Open Programming Standard for Parallel Computing

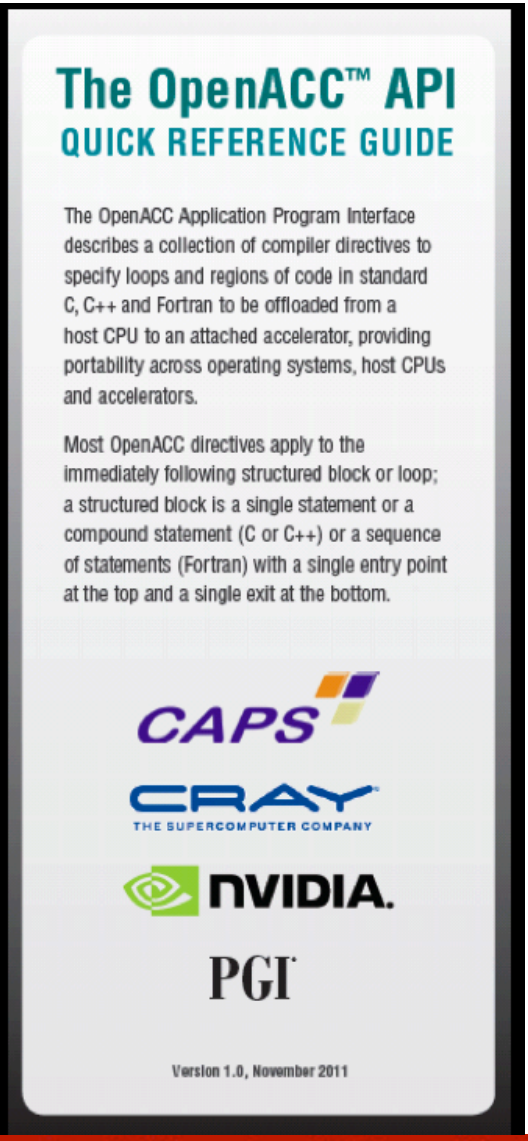
NVIDIA

CRAY THE SUPERCOMPUTER COMPANY

PGI

CAPS

http://www.openacc-standard.org

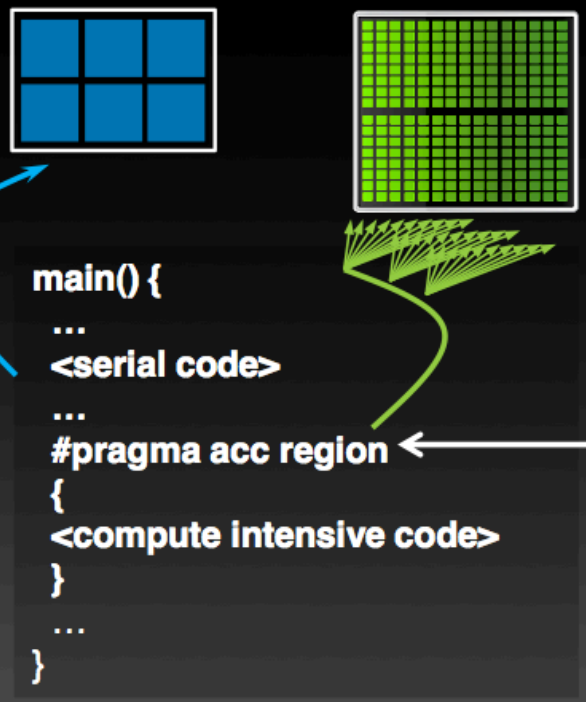


OpenACC is an alternative to computer scientist's CUDA for average programmers

- The idea: Introduce a parallel programming standard for accelerators based on directives (like OpenMP), which:
  - Are inserted into C, C++ or Fortran programs to direct the compiler to parallelize certain code sections.
  - Provide a common code base: Multi-platform and multi-vendor. Enhance portability across other accelerators and multicore CPUs.
  - Bring an ideal way to preserve investment in legacy applications by enabling an easy migration path to accelerated computing.
  - Relax programming effort (and expected performance).
- First supercomputing customers:
  - United States: Oak Ridge National Lab.
  - Europe: Swiss National Supercomputing Centre.

OpenACC: The way it works

Directives: Simple Hints for the Compiler



Compiler Hint

Your original C/Fortran code

Add hints to code

On-ramp to parallel computing

Compiler does heavy lifting of parallelizing code

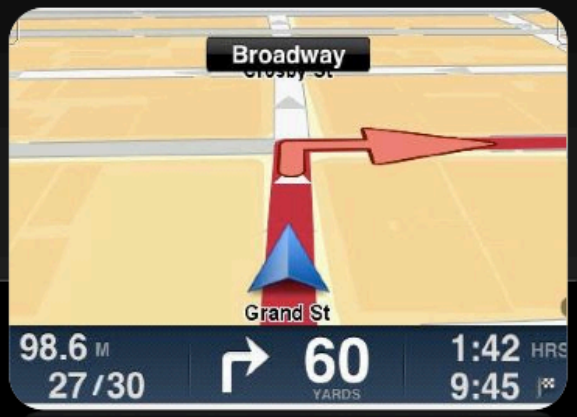
Works on multicore CPUs & many core GPUs

OpenACC: Results

Directives Program Hugely Successful

Real-Time Object Detection

Global Manufacturer of Navigation Systems



5x in 1 Week

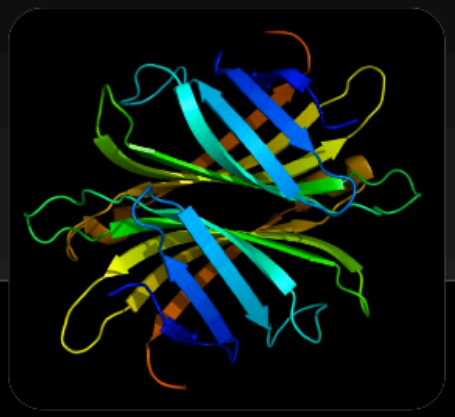
Valuation of Stock Portfolios using Monte Carlo

Global Technology Consulting Company

2x in 4 Hours

Interaction of Solvents and Biomolecules

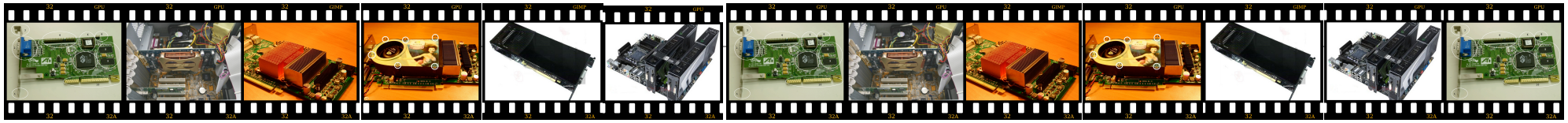
University of Texas at San Antonio

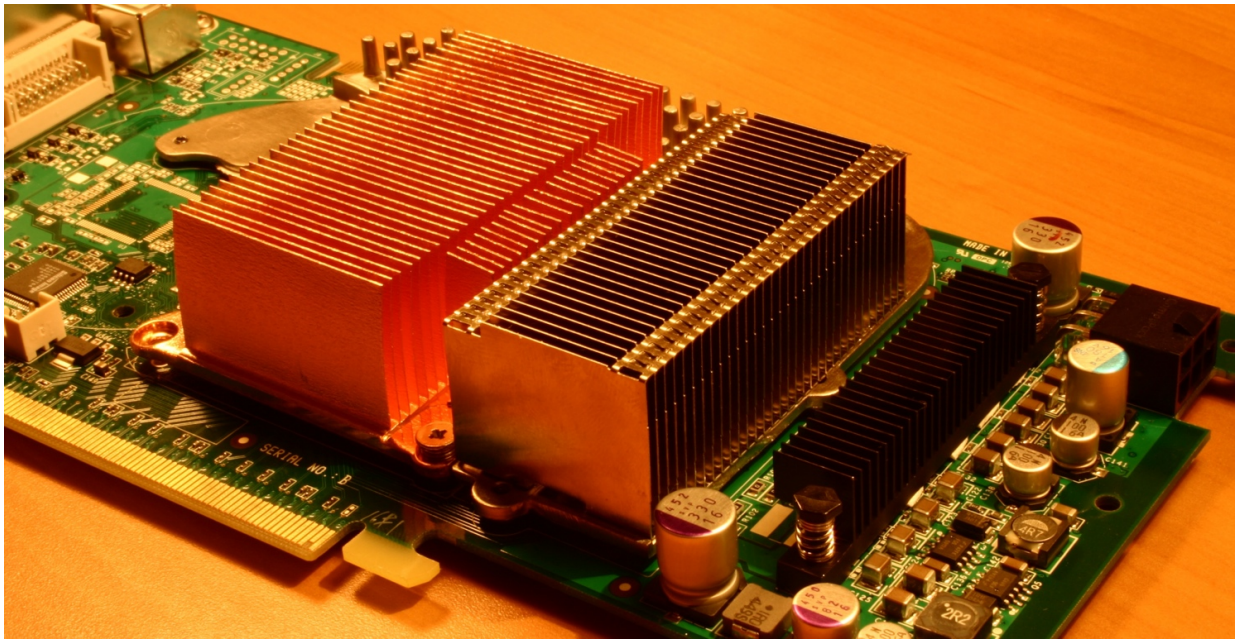


5x in 1 Day

Optimizing code with directives is quite easy, especially compared to CPU threads or writing CUDA kernels. The most important thing is avoiding restructuring of existing code for production applications.

IV. Hardware designs

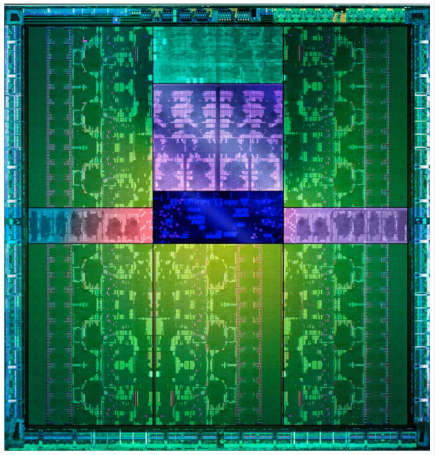




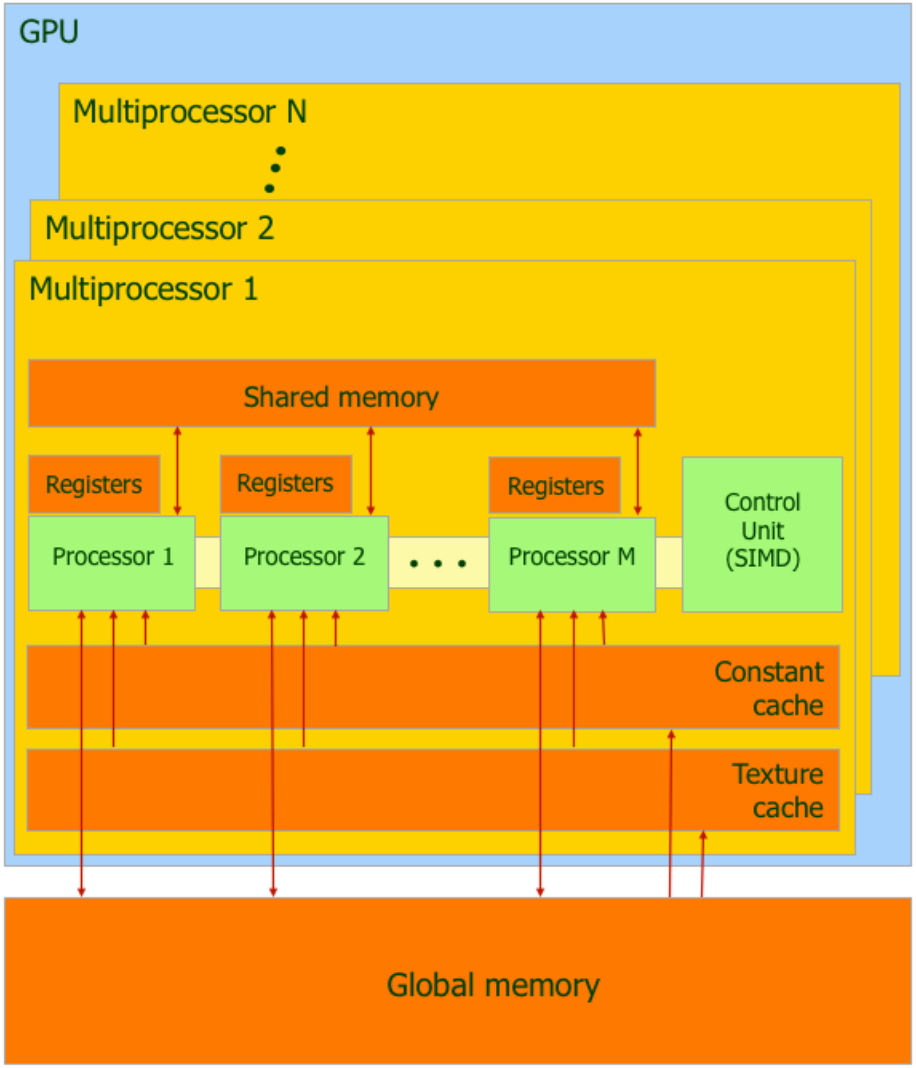
IV. 1. Kepler

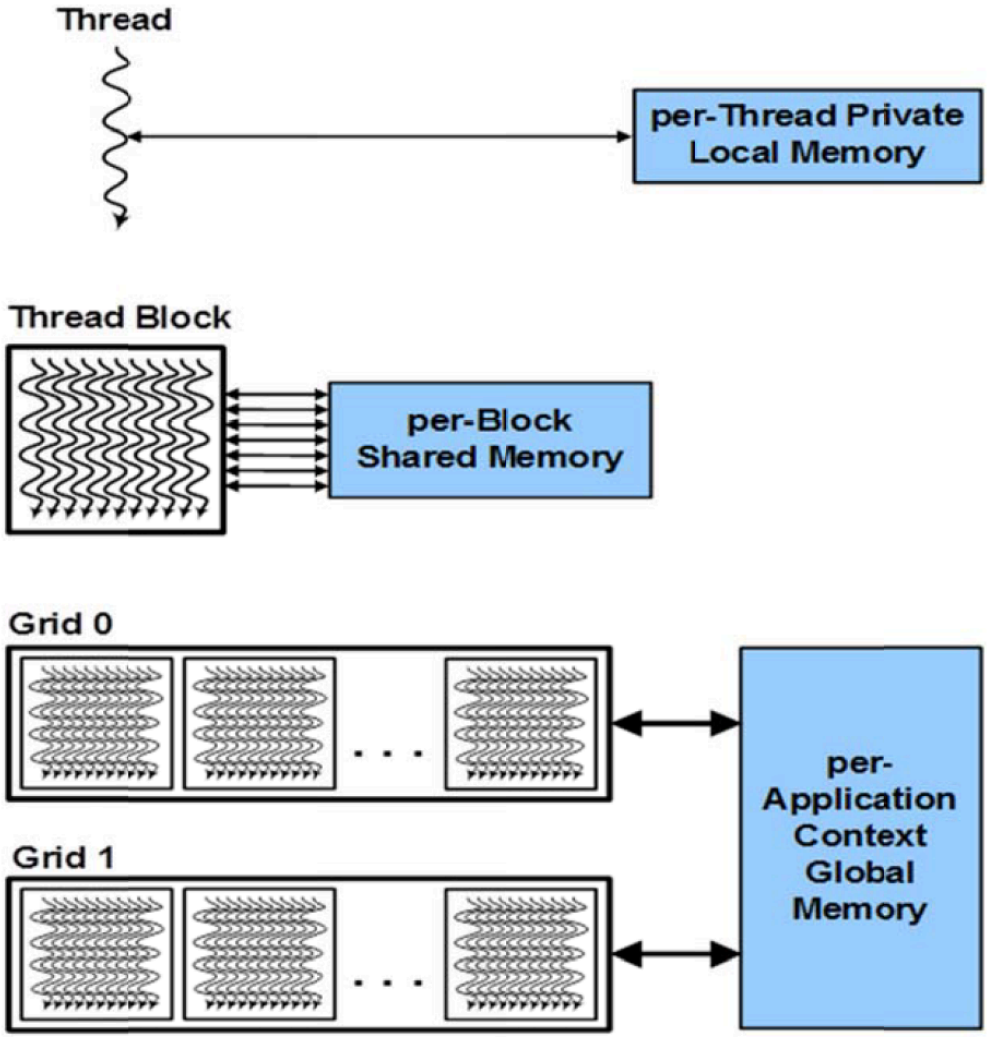
HPC Advisory Council
Spain Conference 2012
September 13th, 2012
University of Malaga
Malaga, Spain

The Kepler architecture: Die and block diagram

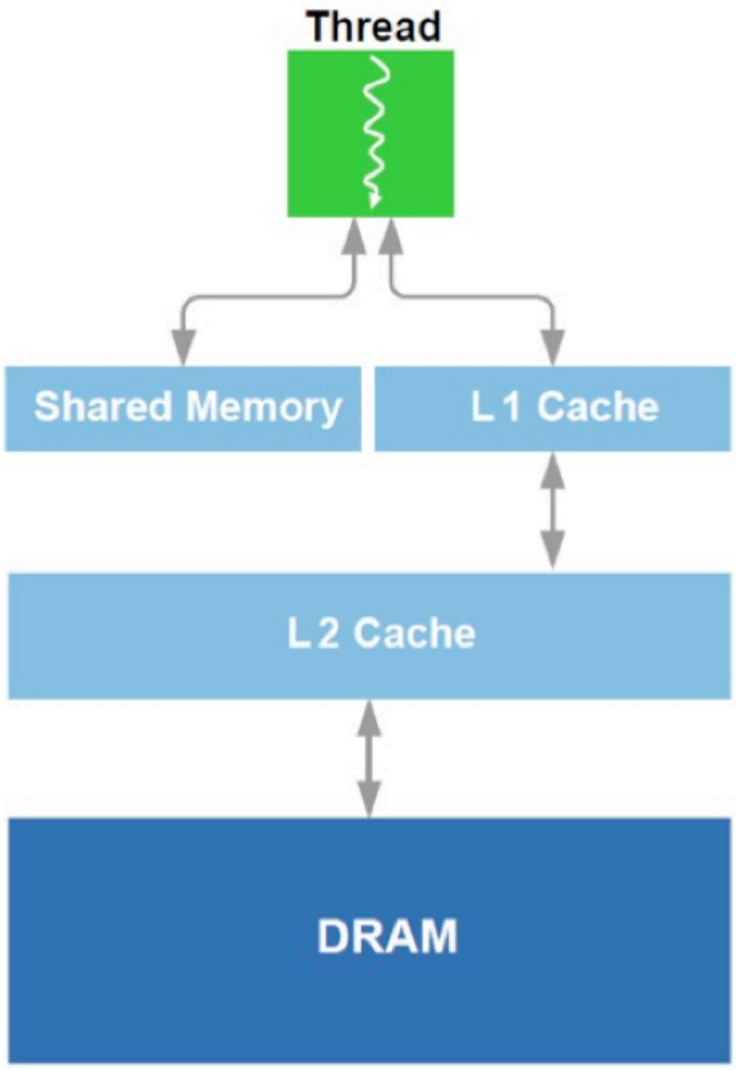


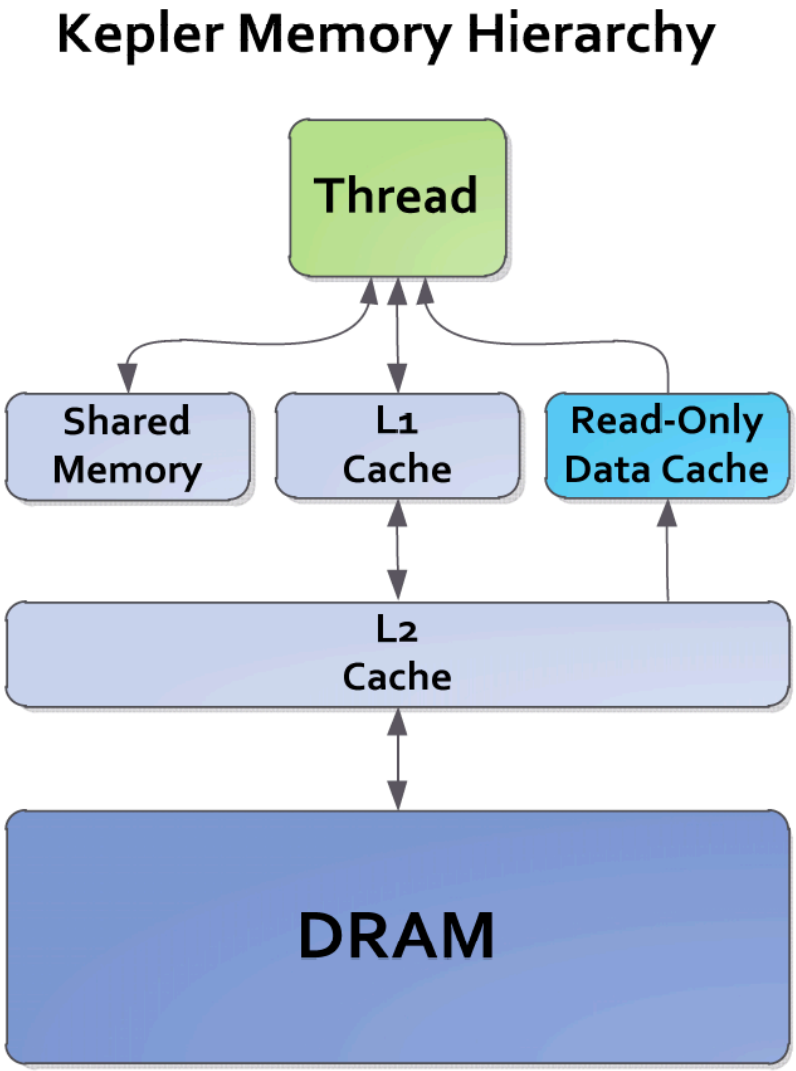
A brief reminder of CUDA





Differences in memory hierarchy





Kepler resources and limitations vs. previous GPU generation

GPU generation	Fermi		Kepler		Limitation	Impact
Hardware model	GF100	GF104	GK104	GK110		
Compute Capability (CCC)	2.0	2.1	3.0	3.5		
Máx. cores (multiprocessors)	512(16)	336(7)	1536(8)	2880(15)	Hardware	Scalability

Kepler resources and limitations vs. previous GPU generation

GPU generation	Fermi		Kepler		Limitation	Impact
Hardware model	GF100	GF104	GK104	GK110		
Compute Capability (CCC)	2.0	2.1	3.0	3.5		
Máx. cores (multiprocessors)	512(16)	336(7)	1536(8)	2880(15)	Hardware	Scalability
Cores / Multiprocessor	32	48	192	192	Hardware	Scalability
Threads / Warp (the warp size)	32	32	32	32	Software	Throughput
Máx. warps / Multiprocessor	48	48	64	64	Software	Throughput
Máx. thread-blocks / Multiproc.	8	8	16	16	Software	Throughput
Máx. threads / Thread-block	1024	1024	1024	1024	Software	Parallelism
Máx. threads / Multiprocessor	1536	1536	2048	2048	Software	Parallelism

Kepler resources and limitations vs. previous GPU generation

GPU generation	Fermi		Kepler		Limitation	Impact
Hardware model	GF100	GF104	GK104	GK110		
Compute Capability (CCC)	2.0	2.1	3.0	3.5		
Máx. cores (multiprocessors)	512(16)	336(7)	1536(8)	2880(15)	Hardware	Scalability
Cores / Multiprocessor	32	48	192	192	Hardware	Scalability
Threads / Warp (the warp size)	32	32	32	32	Software	Throughput
Máx. warps / Multiprocessor	48	48	64	64	Software	Throughput
Máx. thread-blocks / Multiproc.	8	8	16	16	Software	Throughput
Máx. threads / Thread-block	1024	1024	1024	1024	Software	Parallelism
Máx. threads / Multiprocessor	1536	1536	2048	2048	Software	Parallelism
Max. 32-bit registers / thread	63	63	63	255	Software	Working set
32-bit registers / Multiprocessor	32 K	32 K	64 K	64 K	Hardware	Working set
Shared memory / Multiprocessor	16-48 K	16-48K	16-32-48K	16-32-48K	Hardware	Working set

Kepler resources and limitations vs. previous GPU generation

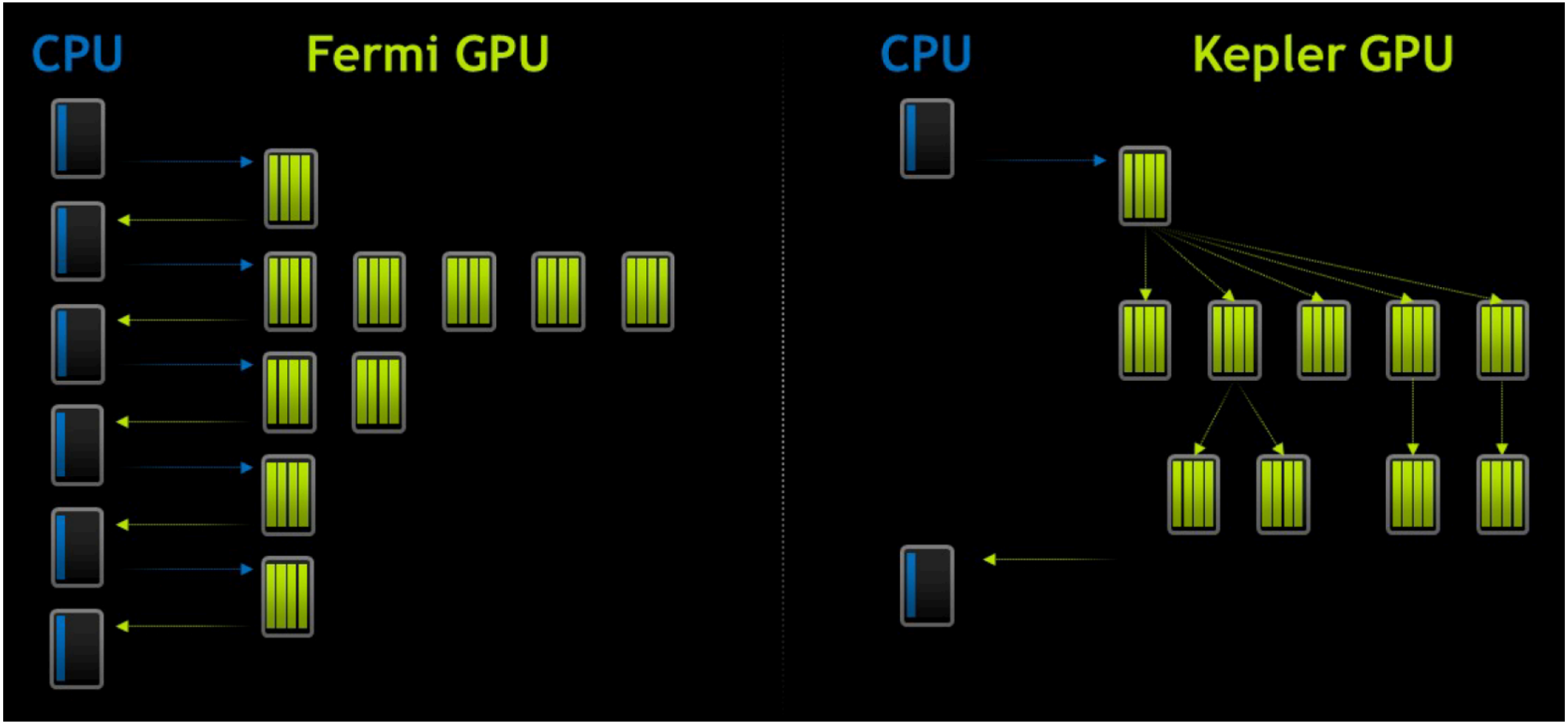
GPU generation	Fermi		Kepler		Limitation	Impact
Hardware model	GF100	GF104	GK104	GK110		
Compute Capability (CCC)	2.0	2.1	3.0	3.5		
Máx. cores (multiprocessors)	512(16)	336(7)	1536(8)	2880(15)	Hardware	Scalability
Cores / Multiprocessor	32	48	192	192	Hardware	Scalability
Threads / Warp (the warp size)	32	32	32	32	Software	Throughput
Máx. warps / Multiprocessor	48	48	64	64	Software	Throughput
Máx. thread-blocks / Multiproc.	8	8	16	16	Software	Throughput
Máx. threads / Thread-block	1024	1024	1024	1024	Software	Parallelism
Máx. threads / Multiprocessor	1536	1536	2048	2048	Software	Parallelism
Max. 32-bit registers / thread	63	63	63	255	Software	Working set
32-bit registers / Multiprocessor	32 K	32 K	64 K	64 K	Hardware	Working set
Shared memory / Multiprocessor	16-48 K	16-48K	16-32-48K	16-32-48K	Hardware	Working set
Máx. X Grid Dimension	2^16-1	2^16-1	2^32-1	2^32-1	Software	Problem size

Kepler resources and limitations vs. previous GPU generation

GPU generation	Fermi		Kepler		Limitation	Impact
Hardware model	GF100	GF104	GK104	GK110		
Compute Capability (CCC)	2.0	2.1	3.0	3.5		
Máx. cores (multiprocessors)	512(16)	336(7)	1536(8)	2880(15)	Hardware	Scalability
Cores / Multiprocessor	32	48	192	192	Hardware	Scalability
Threads / Warp (the warp size)	32	32	32	32	Software	Throughput
Máx. warps / Multiprocessor	48	48	64	64	Software	Throughput
Máx. thread-blocks / Multiproc.	8	8	16	16	Software	Throughput
Máx. threads / Thread-block	1024	1024	1024	1024	Software	Parallelism
Máx. threads / Multiprocessor	1536	1536	2048	2048	Software	Parallelism
Max. 32-bit registers / thread	63	63	63	255	Software	Working set
32-bit registers / Multiprocessor	32 K	32 K	64 K	64 K	Hardware	Working set
Shared memory / Multiprocessor	16-48 K	16-48K	16-32-48K	16-32-48K	Hardware	Working set
Máx. X Grid Dimension	2^16-1	2^16-1	2^32-1	2^32-1	Software	Problem size
Dynamic Parallelism	No	No	No	**Yes**	Hardware	" structure
Hyper-Q	No	No	No	**Yes**	Hardware	T. scheduling

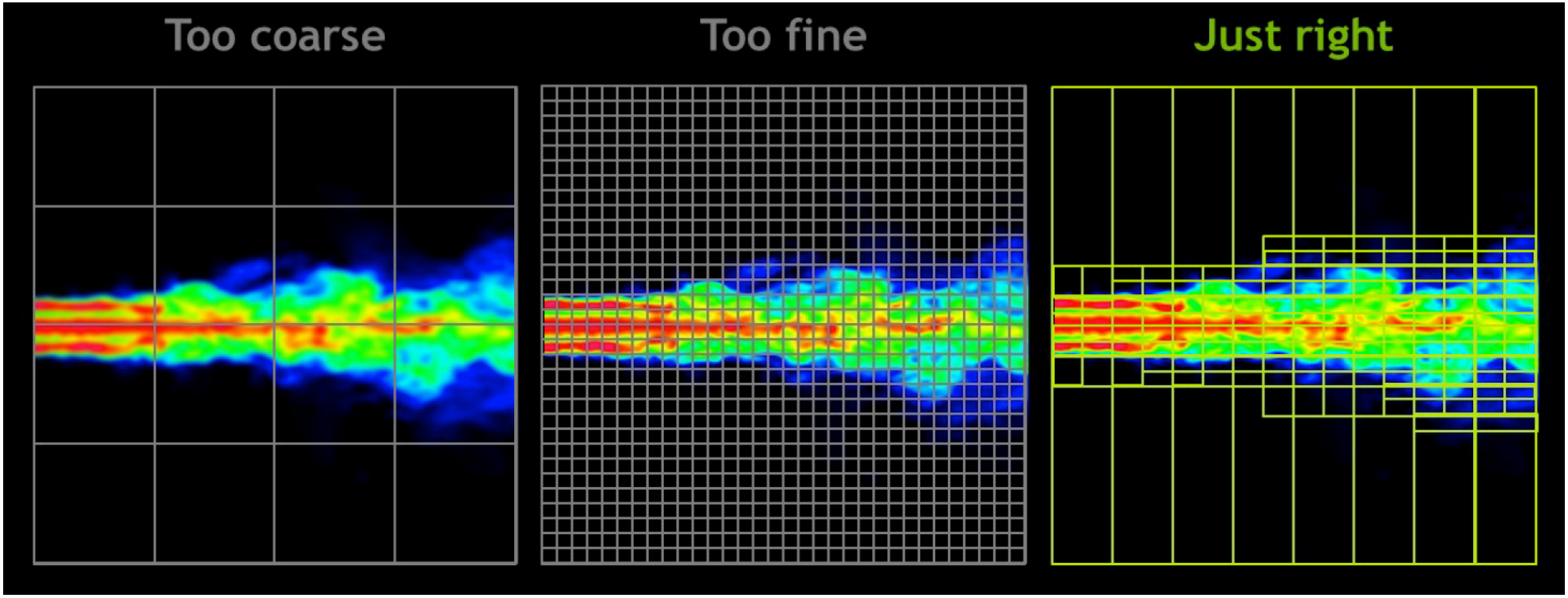
Dynamic Parallelism in Kepler

- Kepler GPUs adapt dynamically to data, launching new threads at run-time.



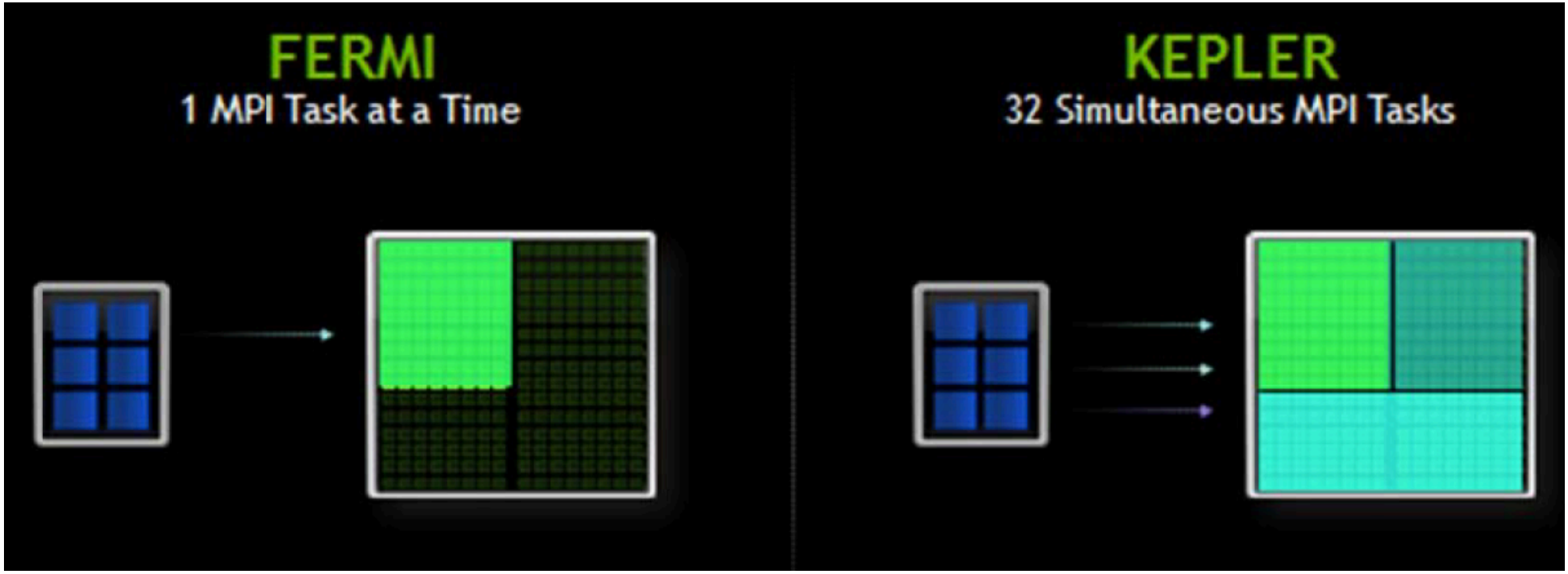
Dynamic Parallelism (2)

- It makes GPU computing easier and broadens reach.

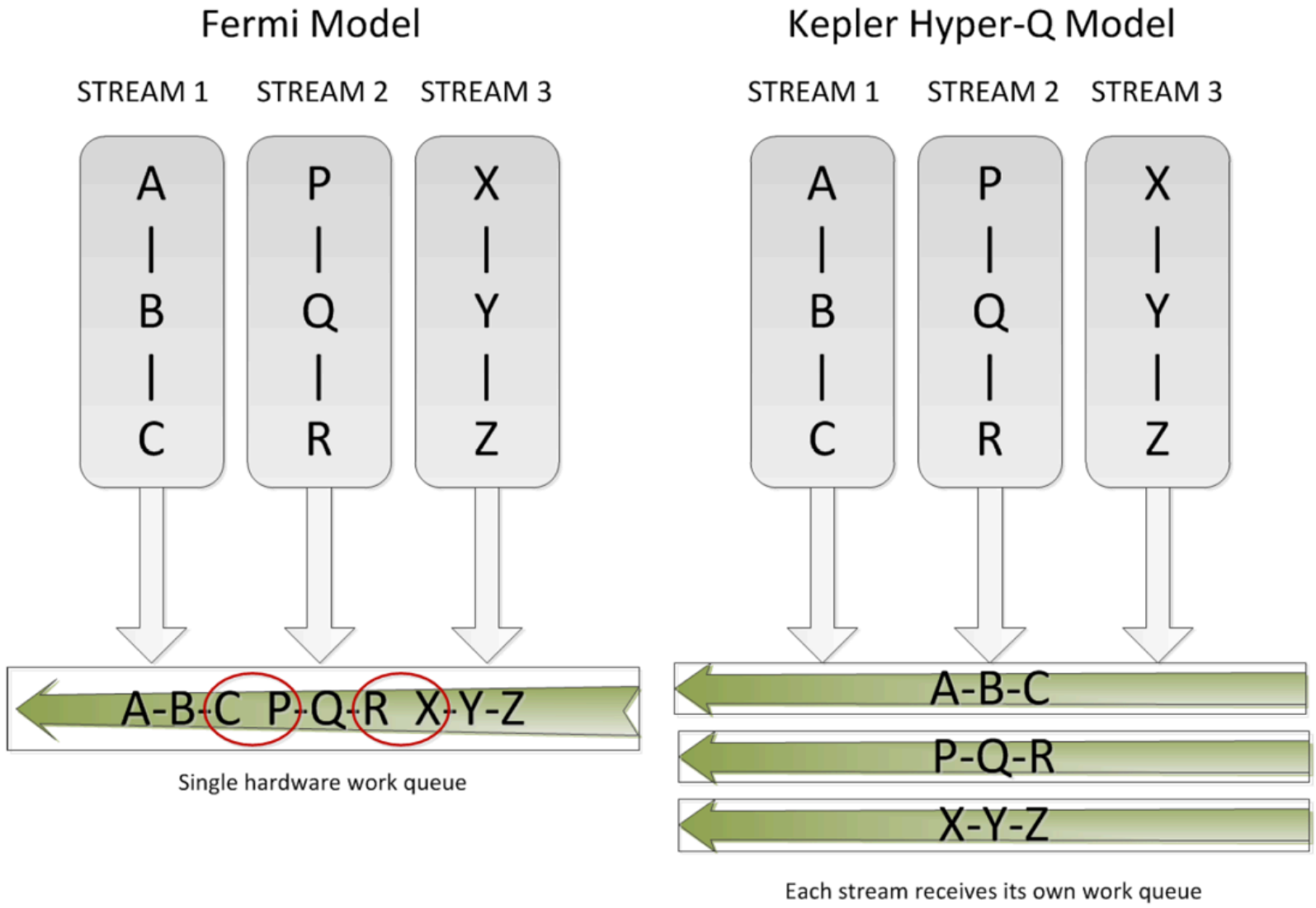


Hyper-Q

- CPU cores simultaneously run tasks on Kepler



Hyper-Q (cont.)

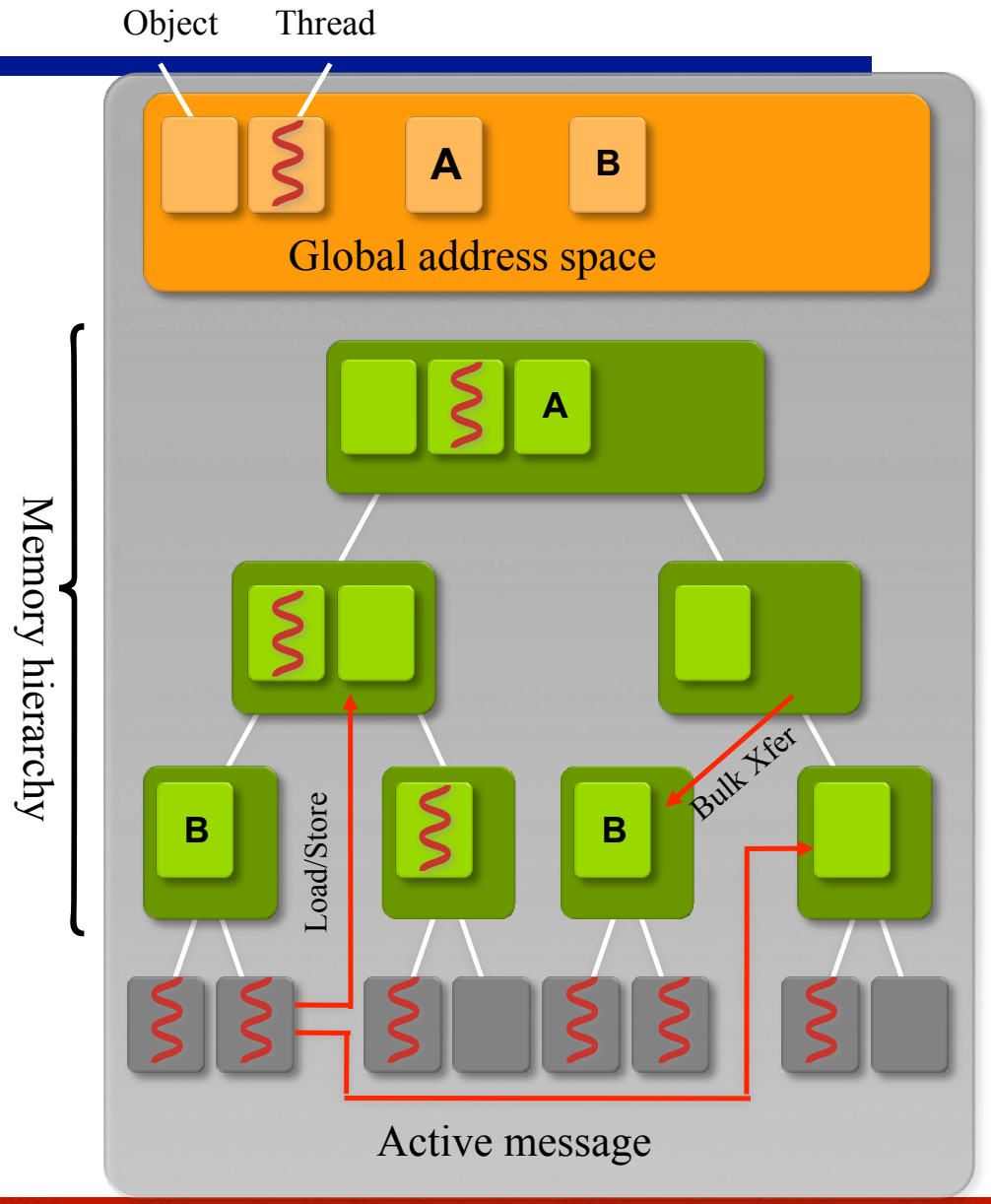

Fermi Model
STREAM 1
STREAM 2
STREAM 3
A
B
C
P
Q
R
X
Y
Z
A-B-C P-Q-R X-Y-Z
Single hardware work queue
Kepler Hyper-Q Model
STREAM 1
STREAM 2
STREAM 3
A
B
C
P
Q
R
X
Y
Z
A-B-C
P-Q-R
X-Y-Z
Each stream receives its own work queue

IV. 2. Echelon

HPC Advisory Council
Spain Conference 2012
September 13th, 2012
University of Malaga
Malaga, Spain

A look ahead: The Echelon execution model

- Swift operations:
  - Thread array creation.
  - Messages.
  - Block transfers.
  - Collective operations.



Thanks for your attention!

- My coordinates:
  - email: ujaldon@uma.es
  - My Web page at University of Malaga:
    - http://manuel.ujaldon.es
  - My web page at Nvidia:
    - http://research.nvidia.com/users/manuel-ujaldon