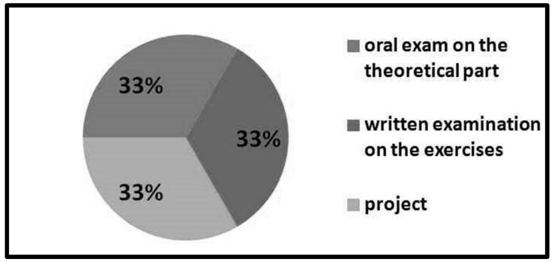
Parallel Systems Course

Introduction

Jan Lemeire
Dept. ETRO
September 23th 2014

Vrije Universiteit Brussel



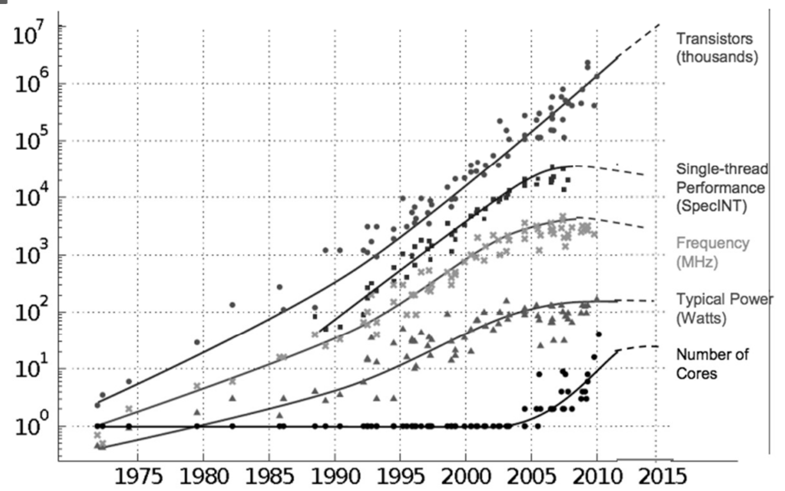
Jan Lemeire (jan.lemeire@vub.ac.be)

- Graduated as Engineer in 1994 at VUB
- Worked for 4 years for 2 IT-consultancy companies
- 2000-2007: PhD at the VUB while teaching as assistant
  - <u>Subject</u>: *probabilistic models for the performance analysis of parallel programs*
- Since 2008: postdoc en parttime professor at VUB, department of electronics and informatics (ETRO)
  - Teaching 'Informatics' for first-year bachelors; 'parallel systems' and 'advanced computer architecture' to masters
- Since October 2012: also teaching for engineers industrial sciences ('industrial engineers')
- Projects, papers, phd students in *parallel processing* (performance analysis, GPU computing) & *data mining/machine learning* (probabilistic models, causality, learning algorithms)
- http://parallel.vub.ac.be

Goals of course

- Understand architecture of modern parallel systems.
- Employ software technologies for parallel programming.
- Design efficient and two-fold generic parallel solutions.
  - For a wide variety of parallel systems & broad class of similar algorithms.
  - Sharpen your low-level and high-level IT skills.
- Understand their performance.
- Make successful technology. Understand economics.

The Free Lunch is Over



Chuck Moore, "DATA PROCESSING IN EXASCALE-CLASS COMPUTER SYSTEMS", The Salishan Conference on High Speed Computing, 2011.

The Free Lunch is Over

Why You Don't Have 10GHz Today?
- **heat/surface is the problem (power wall)**
- **12 ns would mean electric paths of 10 atoms wide**

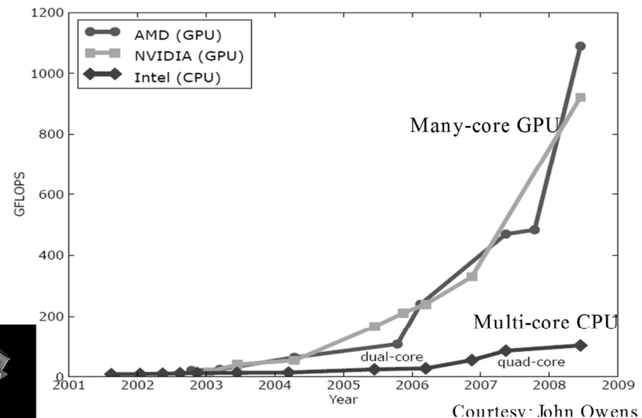
Moreover:
- **memory bottleneck**
- **instruction level parallelism (ILP) wall**

What about Moore's Law?
- **increase of Clock speed: stopped**
- **increase of Transistors: ongoing**

It's now about the number of cores!

http://www.gotw.ca/publications/concurrency-ddj.htm

Floating-Point Operations per Second for the CPU and GPU



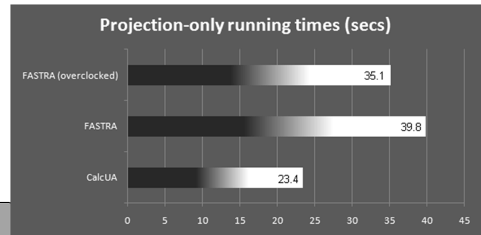
FASTRA at University of Antwerp

http://fastra.ua.ac.be

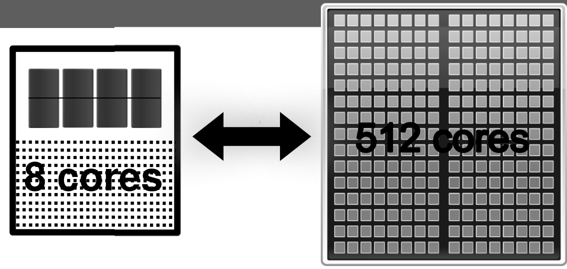
Collection of graphical cards

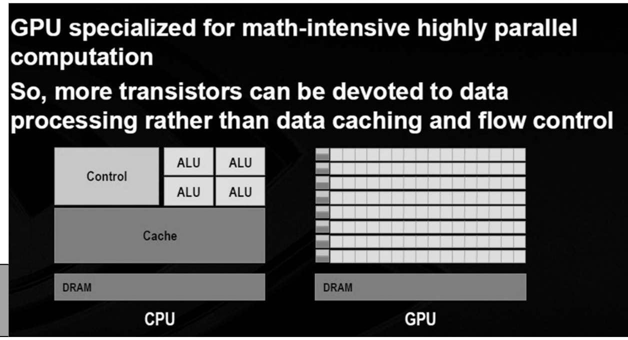
FASTRA 8 cards = 8x128 processors = 4000 euro

Similar performance as University's supercomputer (512 regular desktop PCs) that costed 3.5 million euro in 2005



Why are GPUs faster?





GPU architecture strategy

- Light-weight threads, supported by the hardware
  - Thread processors, upto 96 threads per processor
  - Context switch can happen in 1 cycle!
- No caching mechanism, branch prediction, …
  - GPU does not try to be efficient for every program, does not spend transistors on optimization
  - Simple straight-forward sequential programming should be abandoned…
- Less higher-level memory:
  - GPU: 16KB shared memory per SIMD multiprocessor
  - CPU: L2 cache contains several MB's
- Massively floating-point computation power
- Transparent system organization
  - ⇔ Modern (sequential) CPUs based on simple Von Neumann architecture

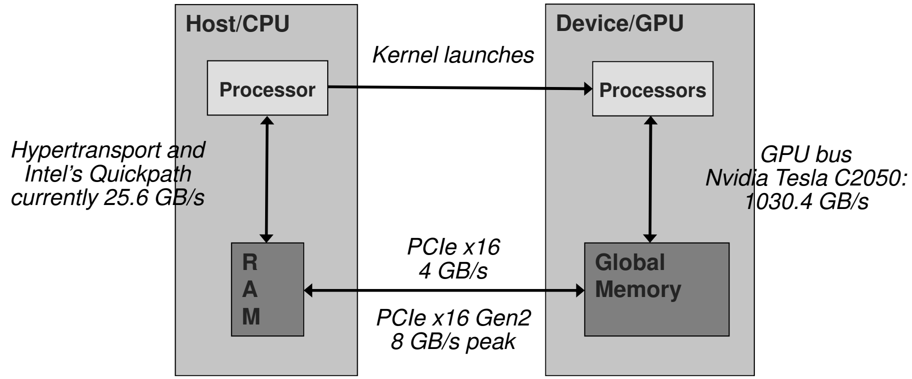
So…

GP-GPUs: Graphics Processing Units for General-Purpose programming

Usage

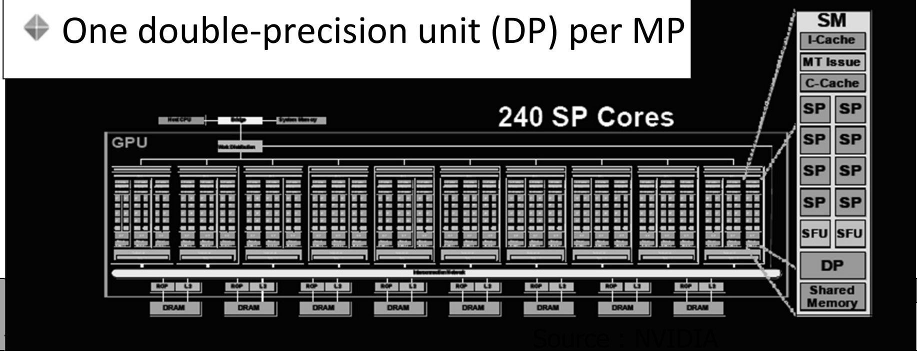
- Copy data from CPU to GPU
- Start kernel within CPU-program (C, java, Matlab, python, …)
  - Several kernels can be launched (pipelined)
  - Handled on the GPU one by one or in parallel
- Figure

Host (CPU) – Device (GPU)



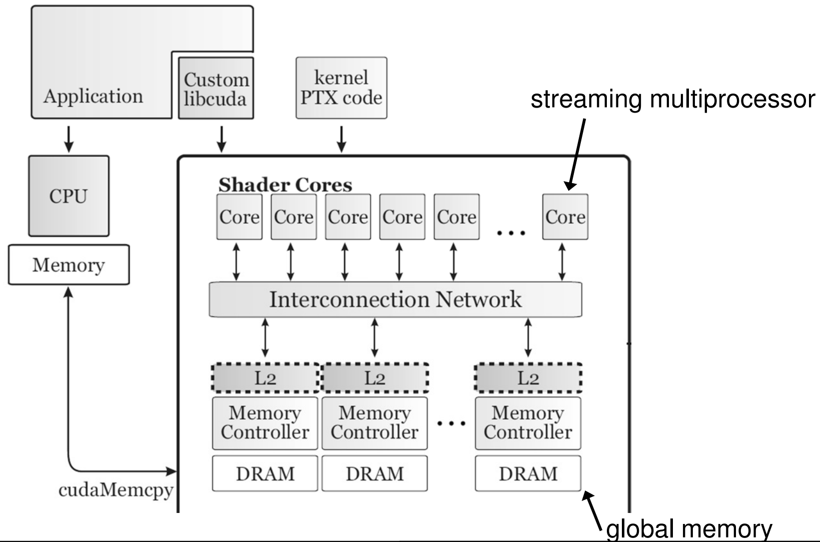
GPU Architecture

- In the GTX 280, there are 10 Thread Processing Clusters
  - Each has 3 Streaming Multiprocessors, which we will refer to as ***multiprocessors (MPs).***
  - Each MP has 8 Thread Processors. We will refer to these as ***Scalar Processors (SP).***
  - **240 processor cores and 30 MPs in total!**
- One double-precision unit (DP) per MP

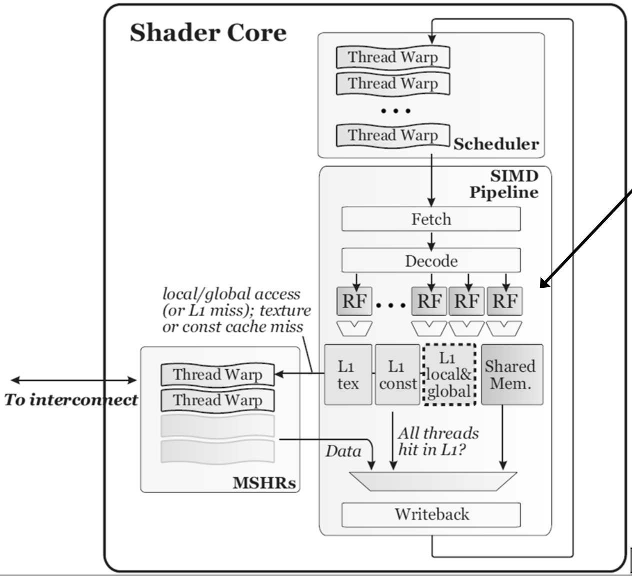


Source : NVIDIA

GPU Architecture



1 Streaming Multiprocessor



The Same Instruction is executed on Multiple Data (SIMD)

width of pipeline: 8 - 32

GPU vs CPU: NVIDIA 280 vs Intel i7 860

	GPU	CPU[1]
Registers	16,384 (32-bit) / multi-processor[3]	128 reservation stations
Peak memory bandwidth	141.7 Gb/sec	21 Gb/sec
Peak GFLOPs	562 (float)/ 77 (double)	50 (double)
Cores	240 (scalar processors)	4/8 (hyperthreaded)
Processor Clock (MHz)	1296	2800
Memory	1Gb	16Gb
Local/shared memory	16Kb/TPC[2]	N/A
[redacted]		

[1]http://ark.intel.com/Product.aspx?id=41316
[2]TPC = Thread Processing Cluster (24 cores)
[3]30 multi-processors in a 280

Performance: GFlops?

- GPUs consist of MultiProcessors (MPs) grouping a number of Scalar Processors (SPs)
- Nvidia GTX 280:
  - 30MPs x 8 SPs/MP x 2FLOPs/instr/SP x 1 instr/clock x 1.3 GHz

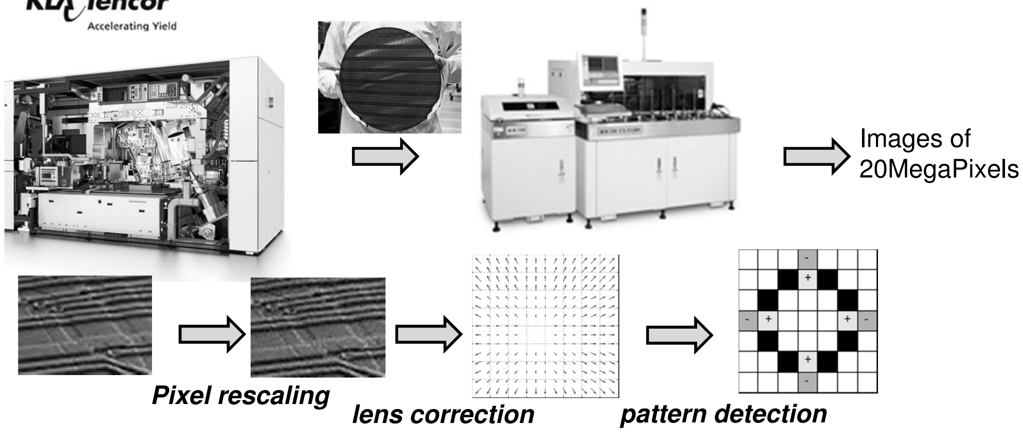
  = 624 GFlops
- Nvidia Tesla C2050:
  - 14 MPs x 32 SPs/MP x 2FLOPs/instr/SP x 1 instr/clock x 1.15 GHz (clocks per second)

  = 1030 GFlops

Other limit: bandwidth

- Nvidia GTX 280:
  - 1.1 GHz memory clock
  - 141 GB/s
- Nvidia Tesla C2050:
  - 1.5 GHz memory clock
  - 144 GB/s

Example: real-time image processing



CPU gives only 4 fps
next generation machines need 50 fps
GPUs deliver 70 fps

Example: pixel transformation

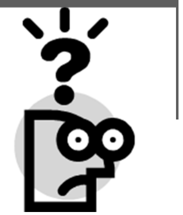
```
usgn_8  transform(usgn_8 in, sgn_16 gain, sgn_16 gain_divide,
sgn_8 offset)
{
   sgn_32 x;

   x = (in * gain / gain_divide) + offset;

   if (x < 0) x = 0;
   if (x > 255) x = 255;
   return x;
 }
```


Pixel transformation

- Performance on Tesla C2050
- 1 pixel is represented by 1 byte [0-255]
- Integer operations: performance is half of floating point operations
- **Two different implementations**:
  - FPN1: 1 pixel per thread
  - FPN4: 4 pixels per thread (treat 4 bytes as 1 ‘word’)

$\mathbf{P_{mem}}$ *(bytes/s)*	115 GB/s	$\mathbf{P_{ops}}$ *(ops/s)*	500 Gops/s
CI *(bytes/pix)*	1/5	*Ops/pix*	5+4 (FPN1) 5+1 (FPN4)
$\mathbf{P_{mem}}$**xCI** *(pix/s)*	**23 Gpix/s**	$\mathbf{P_{ops}}$**/(Ops/pix)**	56 Gpix/s (FPN1) 83 Gpix/s (FPN4)

But… nothing is for free



- Harder to program!
  - Hardware architecture should be taken into account
  - Optimization is important
  - Additional complexity in code
  - Harder to debug, maintain, …
- Algorithms should contain inherently massively fine-grained parallelism