

# GPUs: The Hype, The Reality, and The Future

David Black-Schaffer

Assistant Professor, Department of Information Technology  
Uppsala University

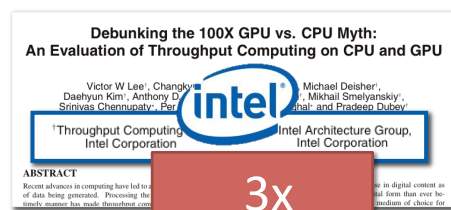
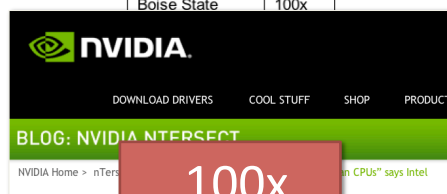
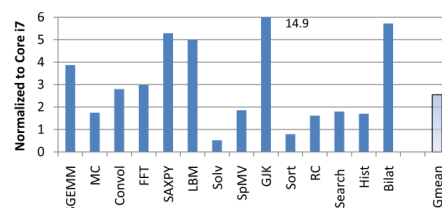
## Today

1. The hype
2. What makes a GPU a GPU?
3. Why are GPUs scaling so well?
4. What are the problems?
5. What's the Future?

## THE HYPE

## How Good are GPUs?


Developer	Speed Up
Massachusetts General Hospital	300x
University of Rochester	160x
University of Amsterdam	150x
Harvard University	130x
University of Pennsylvania	130x
Nanyang Tech, Singapore	130x
University of Illinois	125x
Boise State	100x



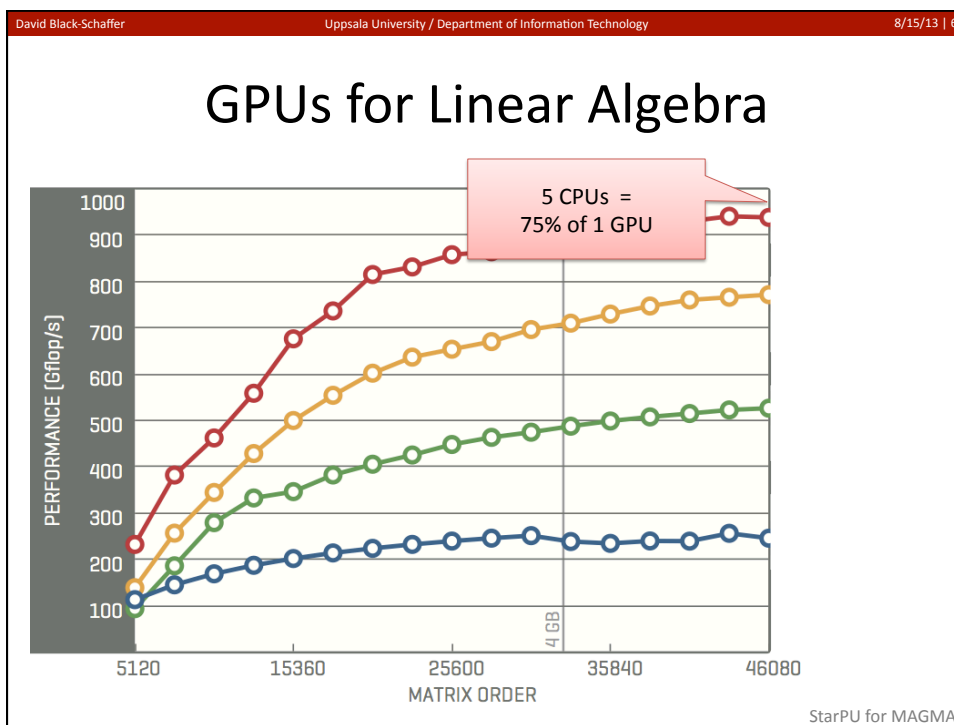
David Black-Schaffer Uppsala University / Department of Information Technology 8/15/13 | 5

## Real World Software

- Press release Nov 2011:
  - “NVIDIA today announced that four leading applications... have added support for multiple GPU acceleration, enabling them to cut simulation times from days to hours.”
- GROMACS
  - **2-3x** overall
  - Implicit solvers **10x**, PME simulations **1x**
- LAMPS
  - **2-8x** for double precision
  - Up to **15x** for mixed
- QMCPACK
  - **3x**



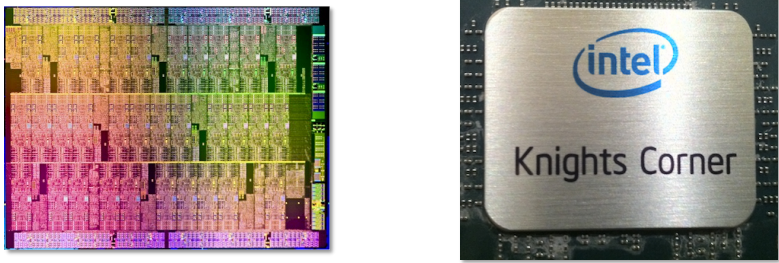
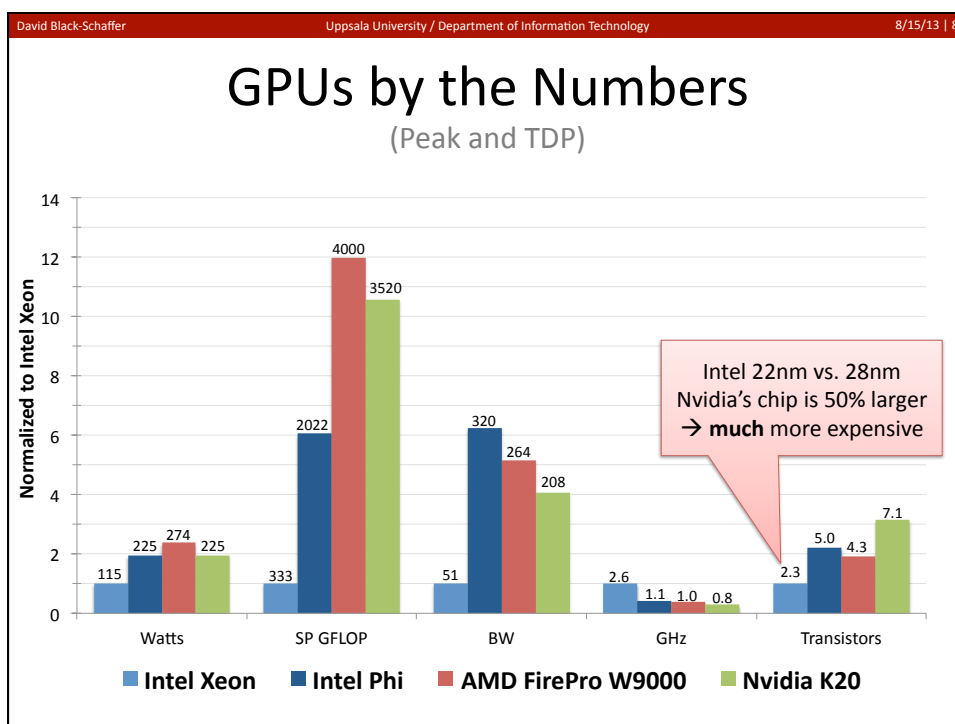
**2x is AWESOME!** Most research claims 5-10%.

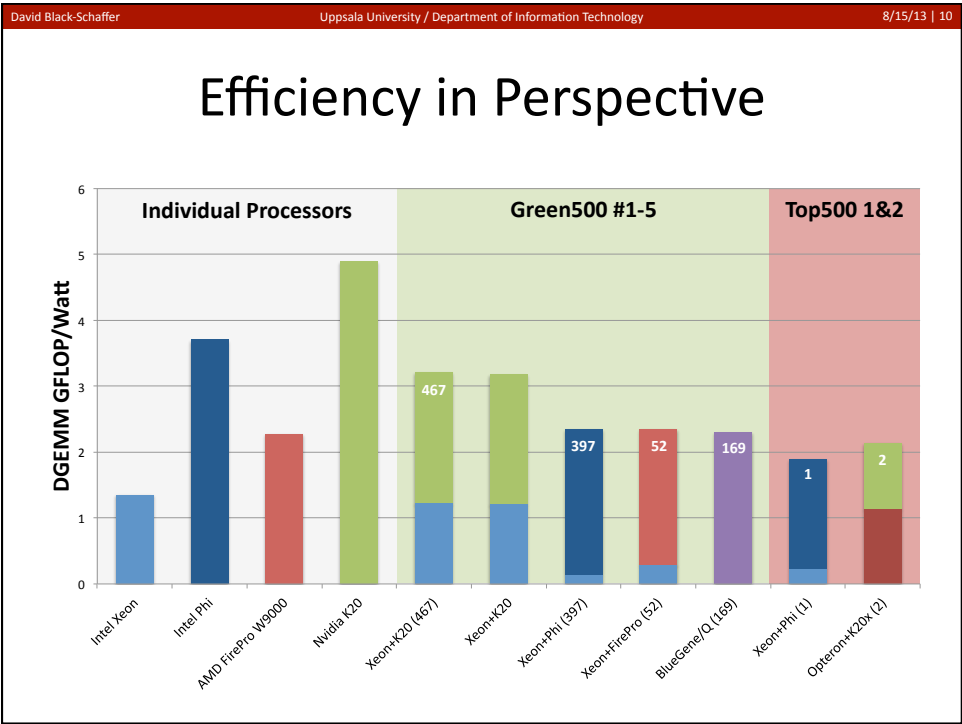
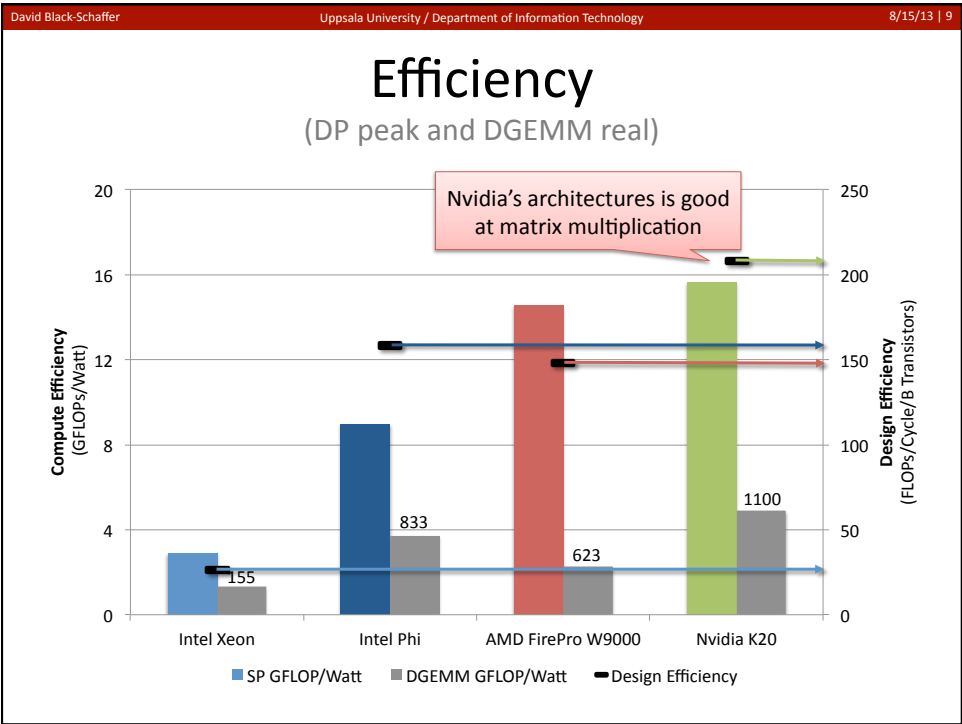


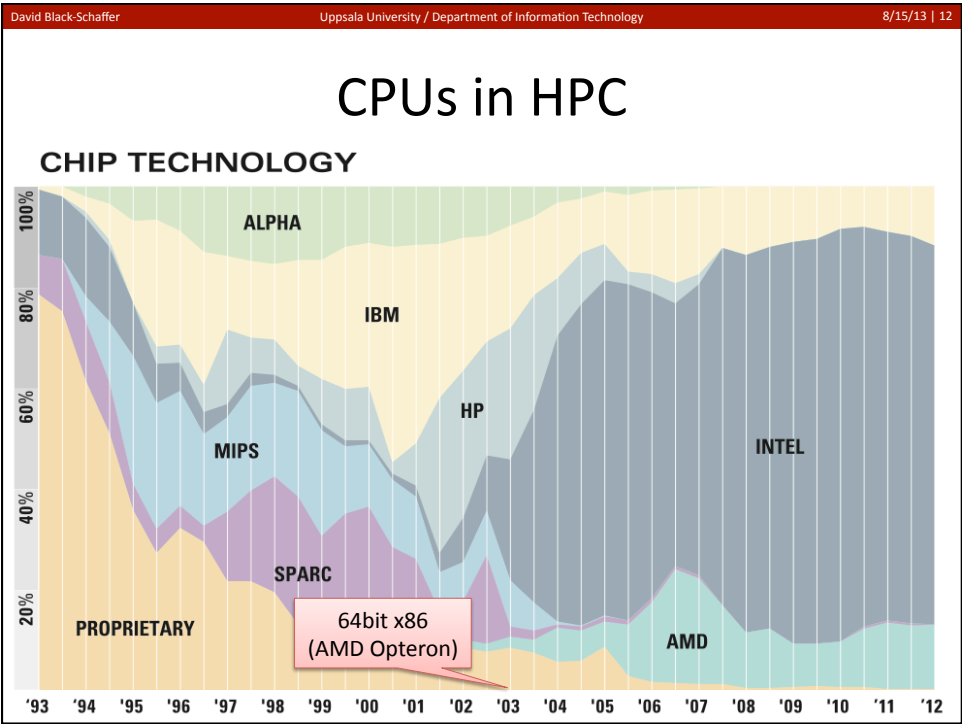
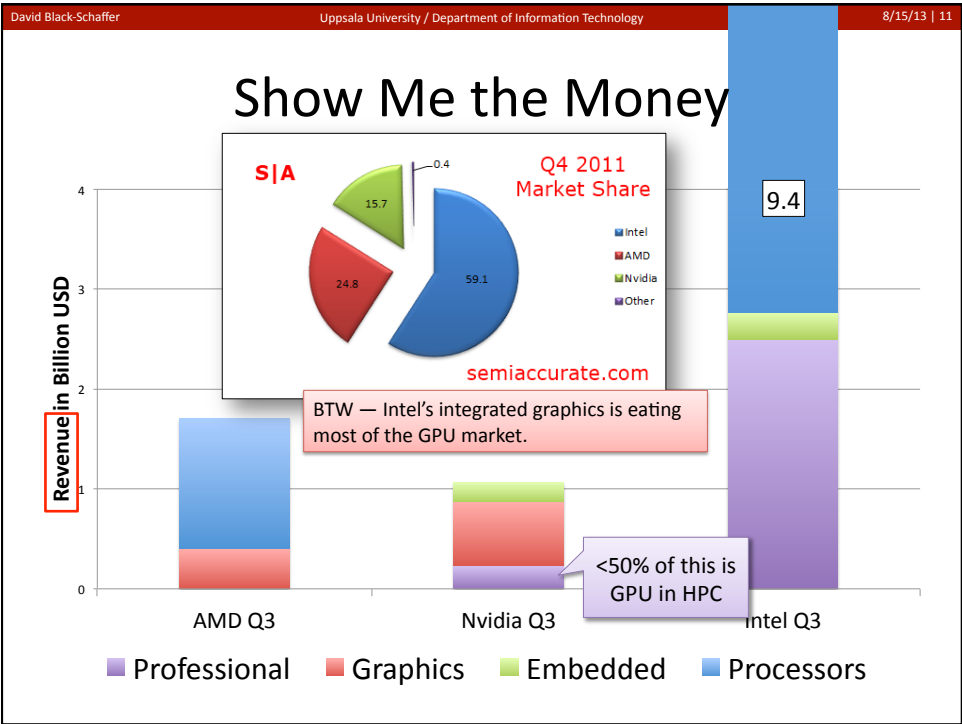
David Black-Schaffer Uppsala University / Department of Information Technology 8/15/13 | 7

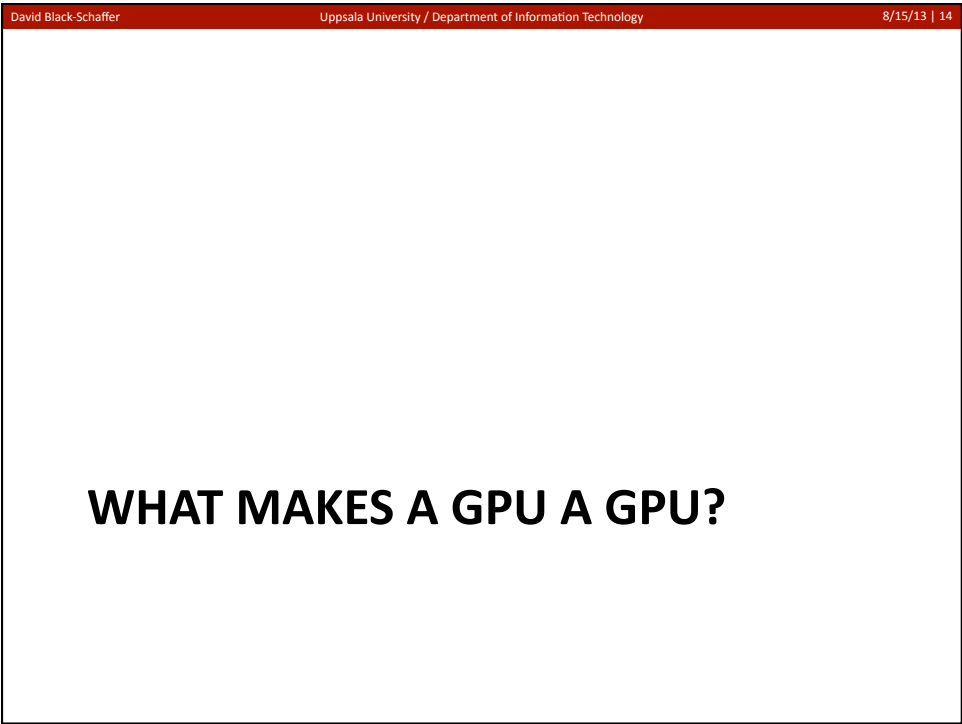
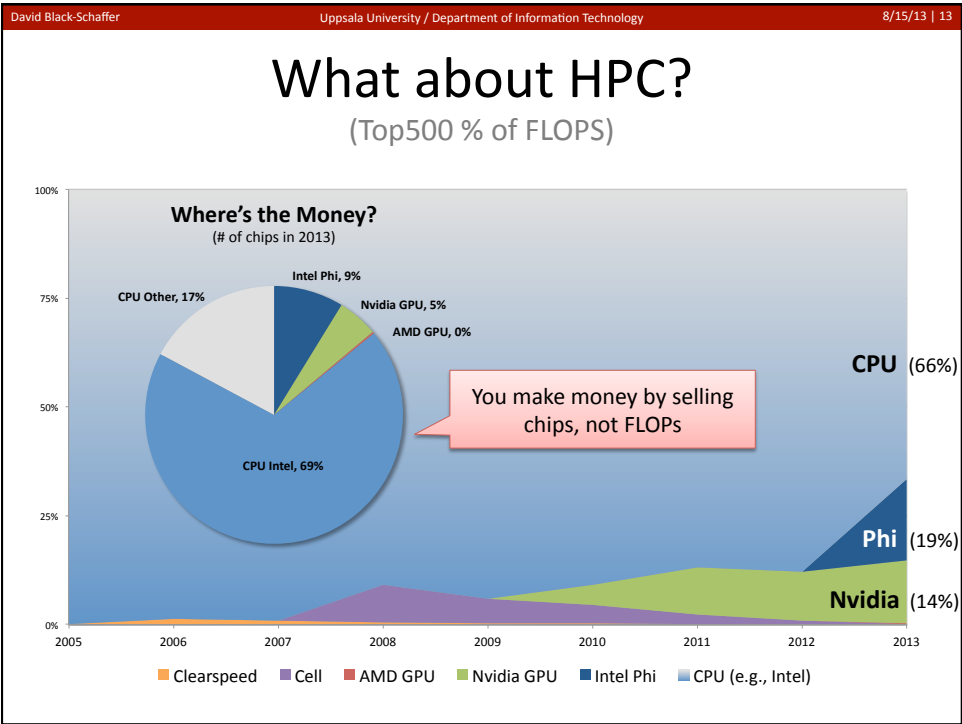
## Intel's Response

- **Larabee**
  - Manycore x86-“light” to compete with Nvidia/AMD in graphics and compute
  - Didn't work out so well (despite huge fab advantages — graphics is hard)
- **Repositioned it as an HPC co-processor**
  - Xeon Phi
  - **1TF double precision** in a **huge** (expensive) single **22nm** chip



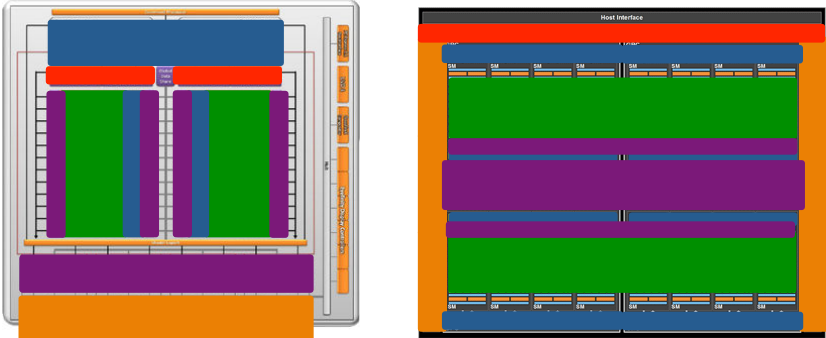




David Black-Schaffer Uppsala University / Department of Information Technology 8/15/13 | 15

## GPU Characteristics

- **Architecture**
  - Data parallel processing
  - Hardware thread scheduling
  - High memory bandwidth
  - Graphics rasterization units
  - Limited caches (with texture filtering hardware)
- **Programming/Interface**
  - Data parallel kernels
  - Throughput-focused
  - Limited synchronization
  - Limited OS interaction



David Black-Schaffer Uppsala University / Department of Information Technology 8/15/13 | 16

## GPU Innovations

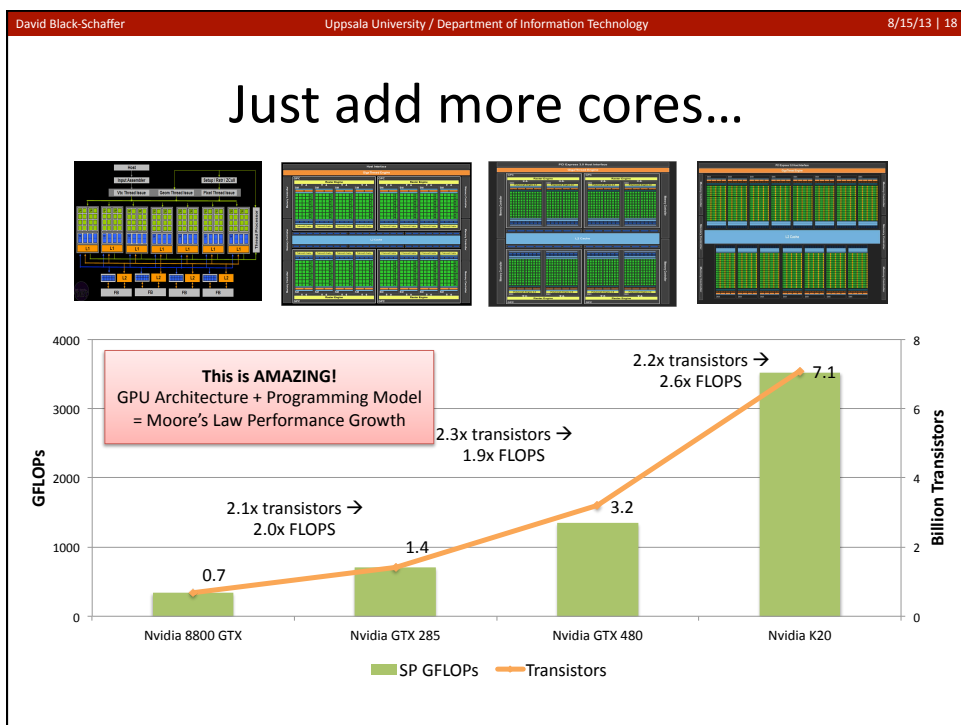
- **SMT (for latency hiding)**
  - Massive numbers of threads
  - Programming SMT is far easier than SIMD
- **SIMT (thread groups)**
  - Amortize scheduling/control/data access
  - Warps, wavefronts, work-groups, gangs
- **Memory systems optimized for graphics**
  - Special storage formats
  - Texture filtering in the memory system
  - Bank optimizations across threads
- **Limited synchronization**
  - Improves hardware scalability

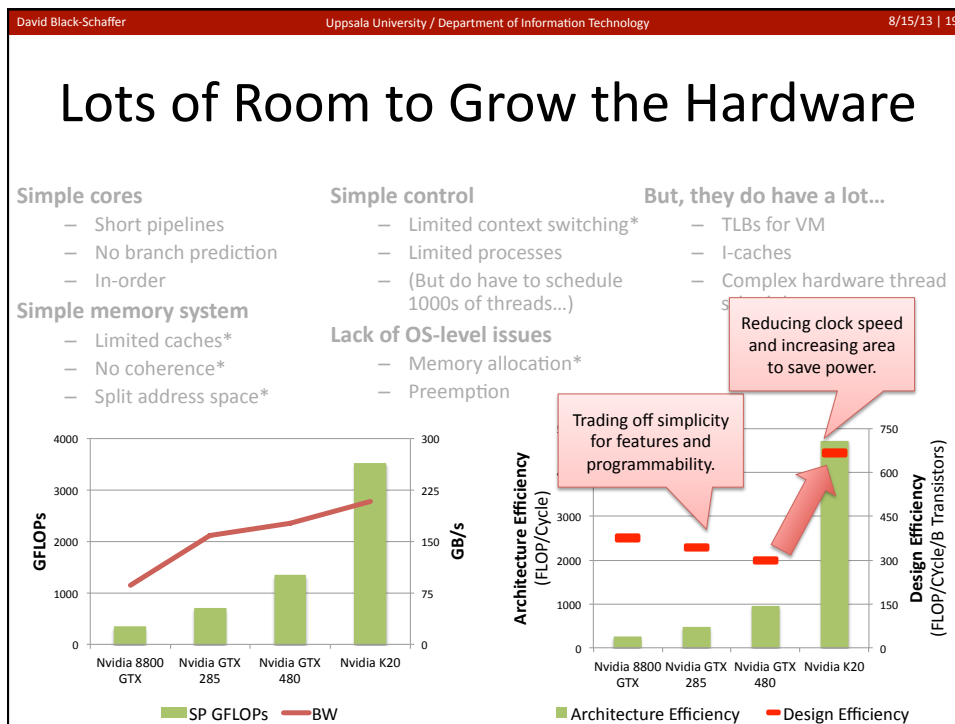
(They didn't invent any of these, but they made them successful.)



David Black-Schaffer Uppsala University / Department of Information Technology 8/15/13 | 17

## WHY ARE GPUS SCALING SO WELL?





David Black-Schaffer Uppsala University / Department of Information Technology 8/15/13 | 20

## “Nice” Programming Model

- All GPU programs have:
  - Explicit parallelism
  - Hierarchical structure
  - Restricted synchronization
  - Data locality
    - Inherent in graphics
    - Enforced in compute by performance
  - Latency tolerance
- **Easy to scale!**

```
void kernel calcSin(global float *data) {
    int id = get_global_id(0);
    data[id] = sin(data[id]);
}
```

**Synchronization OK.**

**No Synchronization.**

**Expensive** (Texture Units, Local Data Storage)

**Cheap** (SIMD Engines)

David Black-SchafferUppsala University / Department of Information Technology8/15/13 | 21

## Why are GPUs Scaling So Well?

- Room in the hardware design
- Scalable software

*They're not burdened with 30 years of cruft and legacy code...*

...lucky them.

David Black-SchafferUppsala University / Department of Information Technology8/15/13 | 22

## WHERE ARE THE PROBLEMS?

David Black-Schaffer Uppsala University / Department of Information Technology 8/15/13 | 23

## Amdahl's Law

- **Always have serial code**
  - GPU single threaded performance is terrible
- **Solution: Heterogeneity**
  - A few **“fat” latency-optimized** cores (CPUs)
  - Many **“thin” throughput-optimized** cores (GPUs)
  - Plus hard-coded accelerators

- **Nvidia Project Denver**
  - ARM for latency
  - GPU for throughput
- **AMD Fusion**
  - x86 for latency
  - GPU for throughput
- **Intel MIC**
  - x86 for latency
  - X86-“light” for throughput

Limits of Amdahl's Law

David Black-Schaffer Uppsala University / Department of Information Technology 8/15/13 | 24




## It's an Accelerator...

- Moving data **to** the GPU is slow...
- Moving data **from** the GPU is slow...
- Moving data **to/from** the GPU is **really** slow.
- Limited data storage
- Limited interaction with OS

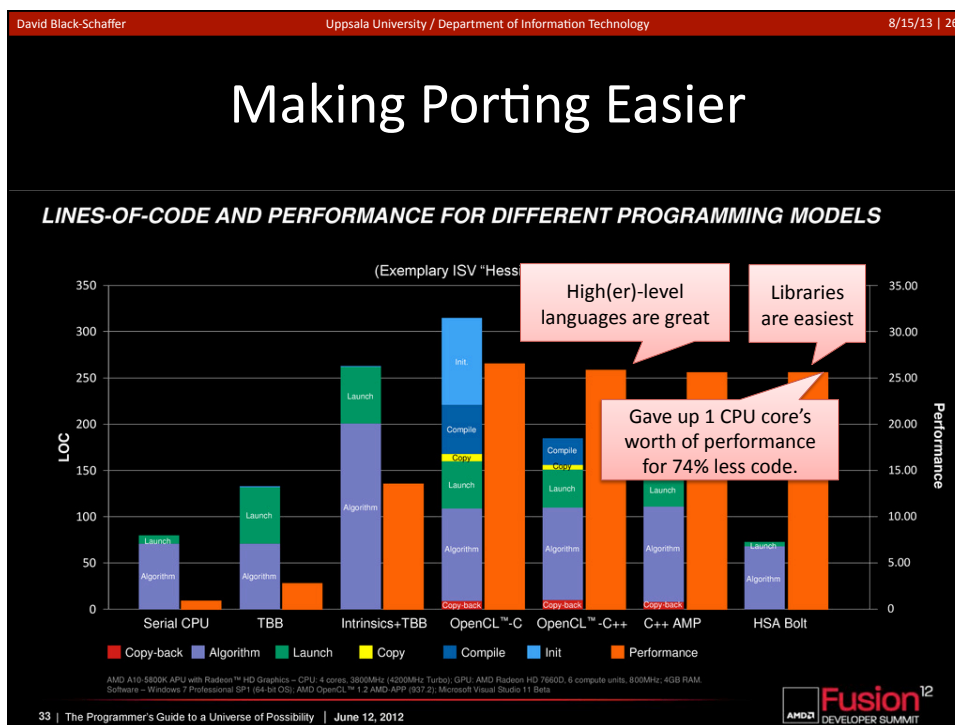
David Black-Schaffer Uppsala University / Department of Information Technology 8/15/13 | 25

## Legacy Code

- **Code lives forever**
  - Amdahl: Even optimizing the 90% of hot code limits speedup to 10x
  - Many won't invest in proprietary technology
- **Programming models are immature**
  - CUDA **mature** **low-level** Nvidia (PGI bought by Nvidia)
  - OpenCL **immature** **low-level** Nvidia, AMD, Intel, ARM, Altera, Apple
  - OpenACC **immature** **high-level** CAPS, Nvidia, Cray, PGI

“...writing either **OpenCL or CUDA** falls into the realm of “**heroic programming**”,  
...**higher level models**...are becoming seen as **increasingly desirable**.”  
—Mark Bull, EPCC



David Black-SchafferUppsala University / Department of Information Technology8/15/13 | 27

## Code that Doesn't Look Like Graphics

- **If it's not painfully data-parallel you have to redesign your algorithm**
  - Example: scan-based techniques for zero counting in JPEG
  - Why? It's only 64 entries!
    - Single-threaded performance is terrible. Need to parallelize.
    - Overhead of transferring data to CPU is too high.
- **If it's not accessing memory well you have to re-order your algorithm**
  - Example: DCT in JPEG
    - Need to make sure your access to local memory has no bank conflicts across threads.
- **Libraries starting to help**
  - Lack of composability
  - Example: Combining linear algebra operations to keep data on the device
- **Most code is not purely data-parallel**
  - Very expensive to synchronize with the CPU (data transfer)
  - No effective support for task-based parallelism
  - No ability to launch kernels from within kernels

David Black-SchafferUppsala University / Department of Information Technology8/15/13 | 28

## Corollary: What are GPUs Good For?

- **Data parallel code**
  - Lots of threads
  - Easy to express parallelism (no SIMD nastiness)
- **High arithmetic intensity and simple control flow**
  - Lots of FPU's
  - No branch predictors
- **High reuse of limited data sets**
  - Very high bandwidth to memory (1-6GB)
  - Extremely high bandwidth to local memory (16-64kB)
- **Code with predictable access patterns**
  - Small (or no) caches
  - User-controlled local memories

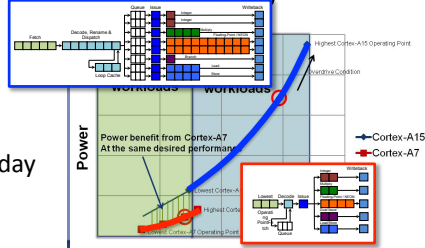
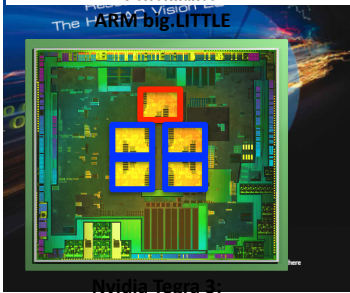
David Black-Schaffer Uppsala University / Department of Information Technology 8/15/13 | 29

# WHAT'S THE FUTURE OF GPUS?

David Black-Schaffer Uppsala University / Department of Information Technology 8/15/13 | 30

## Heterogeneity for Efficiency

- **No way around it in sight**
- **Specialize to get better efficiency**
  - This is why GPUs are more efficient today
- **Heterogeneous mixes**
  - Throughput-oriented “thin” cores
  - Latency-focused “fat” cores
- **Fixed-function accelerators**
  - Video, audio, network, etc.
  - Already in OpenCL 1.2
- **Dark silicon**
  - OS/runtime/app will have to adapt
  - Energy will be a shared resource

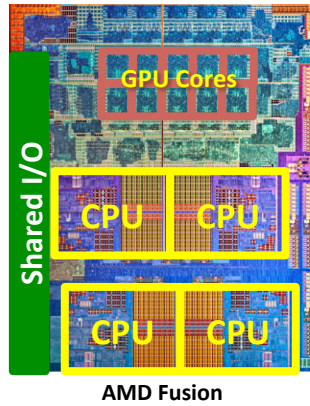
Nvidia Tegra 3:  
4 fast cores + 1 slow core

David Black-Schaffer Uppsala University / Department of Information Technology 8/15/13 | 31

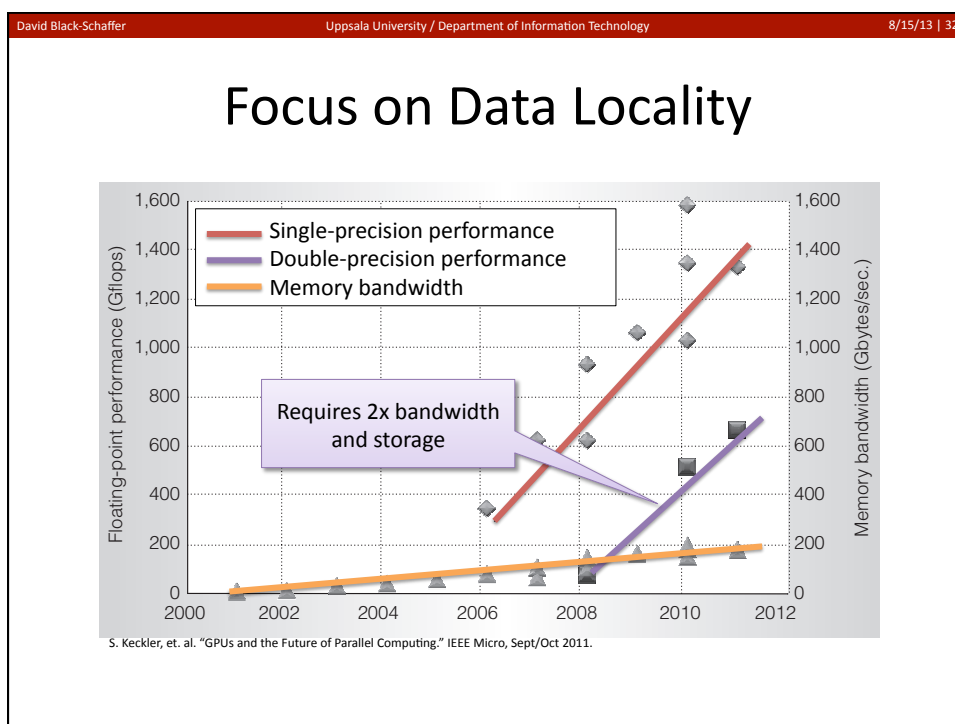
## The Future is Not in Accelerators

- **Memory**
  - Unified memory address space
  - Low performance coherency
  - High performance scratchpads
- **OS interaction between all cores**

- **Nvidia Project Denver**
  - ARM for latency
  - GPU for throughput
- **AMD Fusion**
  - x86 for latency
  - GPU for throughput
- **Intel Phi+Xeon**
  - x86 for latency
  - X86-“light” for throughput



The diagram illustrates the AMD Fusion architecture. It features a central 'GPU Cores' block (red) at the top, connected to two 'CPU' blocks (yellow) below it. A vertical green bar on the left is labeled 'Shared I/O'. The entire structure is labeled 'AMD Fusion' at the bottom.





David Black-SchafferUppsala University / Department of Information Technology8/15/13 | 33

## Focus on Data Locality

- **Not just on-chip/off-chip but *within* a chip**
- **Software controllable memories**
  - Configure for cache/scratch pad
  - *Enable/disable coherency*
  - Programmable DMA/prefetch engines
- **Program must expose data movement/locality**
  - Explicit information to the runtime/compiler
  - Auto-tuning, data-flow, optimization
- **But we will have global coherency to get code correct**

(See the Micro paper “GPUs and the Future of Parallel Computing” from Nvidia about their Echelon project and design.)

Too complex for mortals.  
Need better tools.

David Black-SchafferUppsala University / Department of Information Technology8/15/13 | 34

## Graphics will (still) be a Priority

- It's where the money is
- Fixed-function graphics units
- Memory-system hardware for texture interpolation will live forever...
- Half-precision floating point will live forever...  
(And others else might actually use it. Hint hint.)

David Black-SchafferUppsala University / Department of Information Technology8/15/13 | 35

# CONCLUSIONS

David Black-SchafferUppsala University / Department of Information Technology8/15/13 | 36


## Breaking Through The Hype

- **Real efficiency advantage**
  - Intel is pushing hard to minimize it
  - Much larger for single precision
- **Real performance advantage**
  - About 2-5x
  - But you have to re-write your code (*this is the killer*)
- **The market for GPU compute is small**
  - And Intel is trying to kill it with Phi
- **Everyone** believes that **specialization** is necessary to tackle **energy efficiency**

David Black-SchafferUppsala University / Department of Information Technology8/15/13 | 37

## The Good, The Bad, and The Future

- **The Good:**
  - Limited domain allows more efficient implementation
  - Good choice of domain allows good scaling
- **The Bad:**
  - Limited domain focus makes some algorithms hard
  - They are not x86/linux (legacy code)
- **The Future:**
  - Throughput-cores + latency-cores + fixed accelerators
  - Code that runs well on GPUs today will port well
  - We may share hardware with the graphics subsystem, but we won't "program GPUs"

UPMARC Uppsala Programming for  
Multicore Architectures  
Research Center