Faculty of Computer Science Institute for System Architecture, Operating Systems Group

OS Paper Reading Group
Summer Term 2009

Dresden, 2009-04-07

Administrative Stuff

- Website: www.tudos.org -> Teaching -> Reading Group
- Mailing List: http://os.inf.tu-dresden.de/mailman/listinfo/paper-reading-group

Earning credits

- Each student picks a paper
  - *Explore* field of research by reading the paper and all related work
  - *75 min presentation* on the research topic presented by the paper
  - *8 pages survey paper* on the field of research, summing up everything thoroughly
- Finally: pick an own research topic and write a paper suitable for submission to a (minor) workshop
- Btw.: These are the rules at Johns-Hopkins-University. ;)

TECHNISCHE UNIVERSITÄT DRESDEN

Our modus

- Modus
  - 1 paper each week
    - Related to systems research
    - 10-15 min presentation
      - Paper content
      - Questions regarding the paper / the general topic
    - ***Discussion***
  - Choice of paper
    - *Staff members* propose 3 alternatives, voting on the mailing list.
    - *Students* pick a paper from the list on the web site
    - Decision made on Friday before the next meeting
- Pizza anyone?

Earning credits (2 SWS)

- Send a paper summary to doebel@tudos.org until the day before the presentation (23:59:59)
  - Explain what you understood from the paper.
  - Ask questions about things you did not understand.
  - Mention things you like/dislike about the paper.
  - ...
- Present one paper yourself during the term.
  - English (this is not a test of your language skills!)
  - Show that you understood the paper.
  - Prepare questions for discussion.
  - Extended knowledge (e.g., related work) can be a plus.

Faculty of Computer Science Institute for System Architecture, Operating Systems Group

Larrabee: A many-core x86 architecture for visual computing

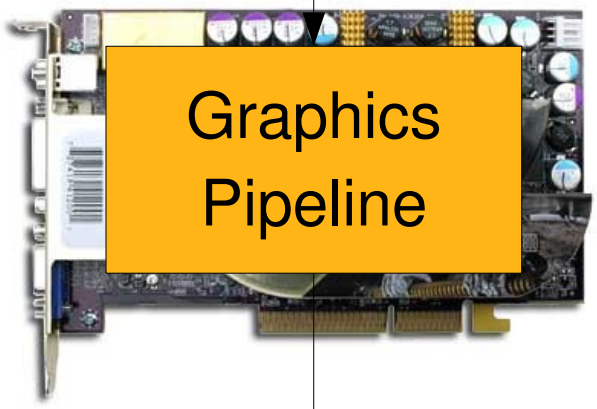
(loads of authors from Intel, RAD, and Stanford)

presented by Bjoern Doebel

Dresden, 2009-04-07

3D Graphics are Complex...

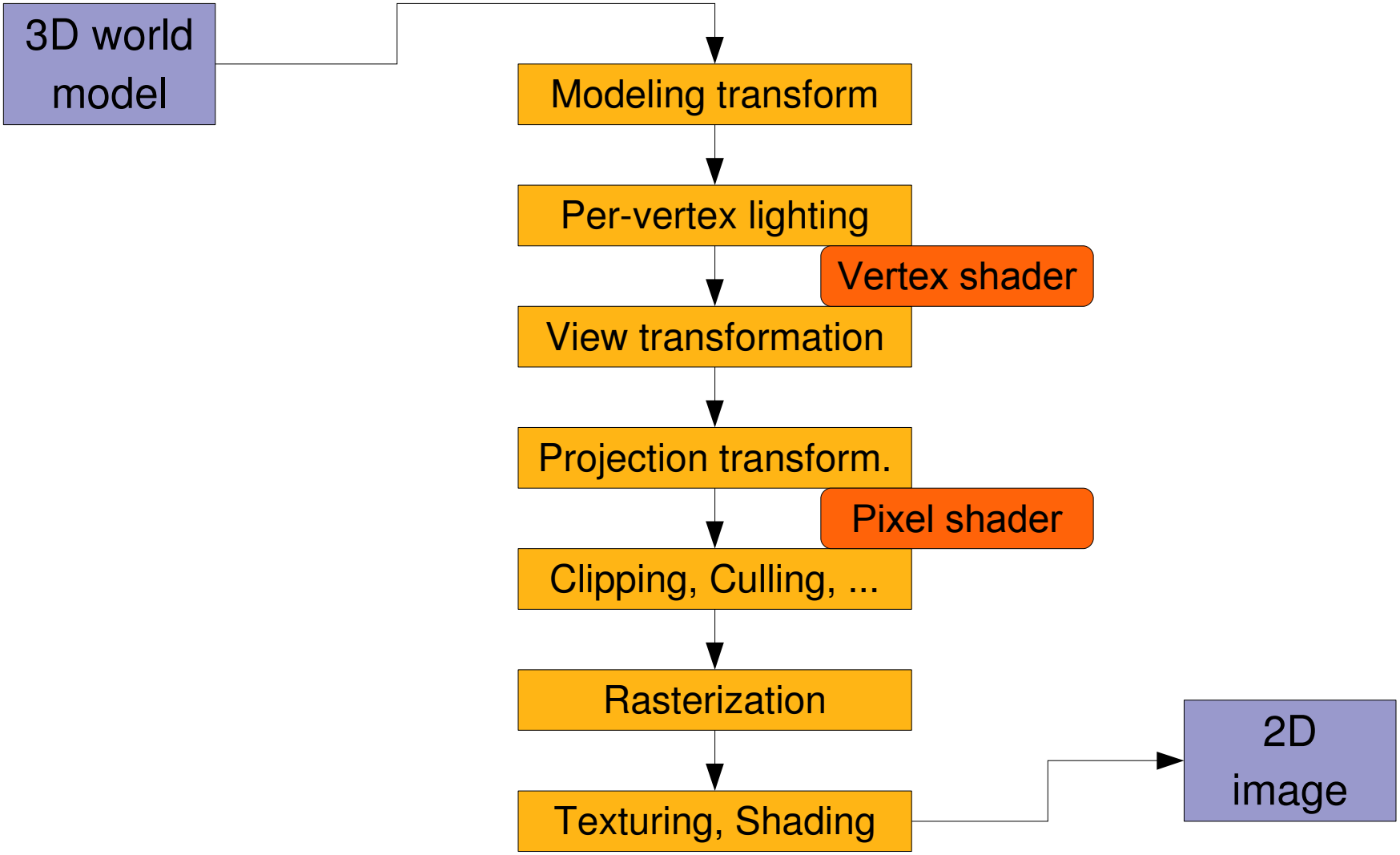
3D world model



OpenGL
DirectX

2D image

3D Graphics are Complex...



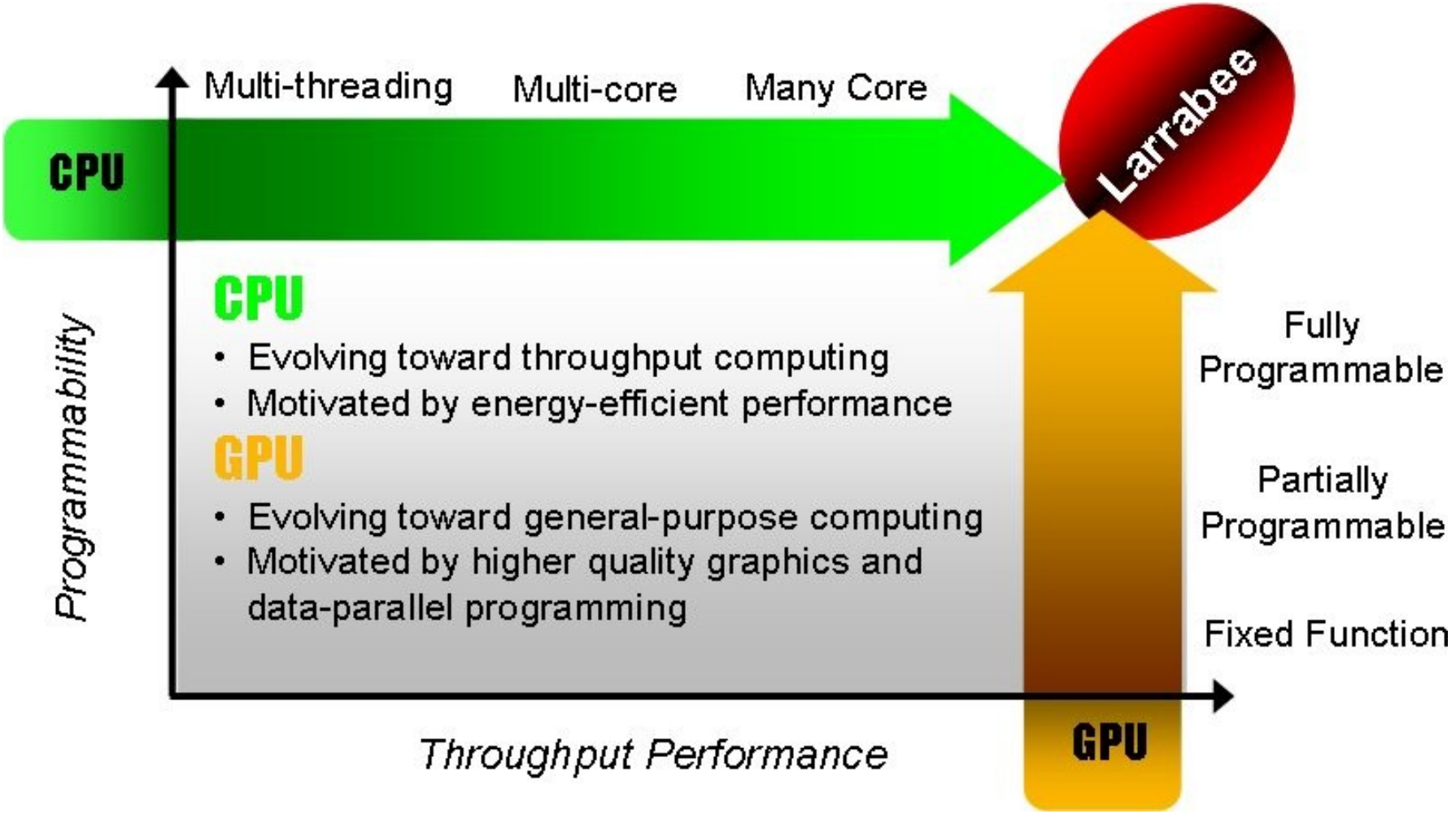
3D Graphics are Highly Parallelizable

- Operations work on
  - Single vertices
  - Groups of vertices
  - Tiles / Bins of the image
- Suitable for SIMD architectures
  - Multiple GPUs per graphics card
  - Local GPU caches, vector processing
- Swap point of view: use GPU for arbitrary computations – stream processing
  - CUDA (nVIDIA), FireStream (AMD/ATI)

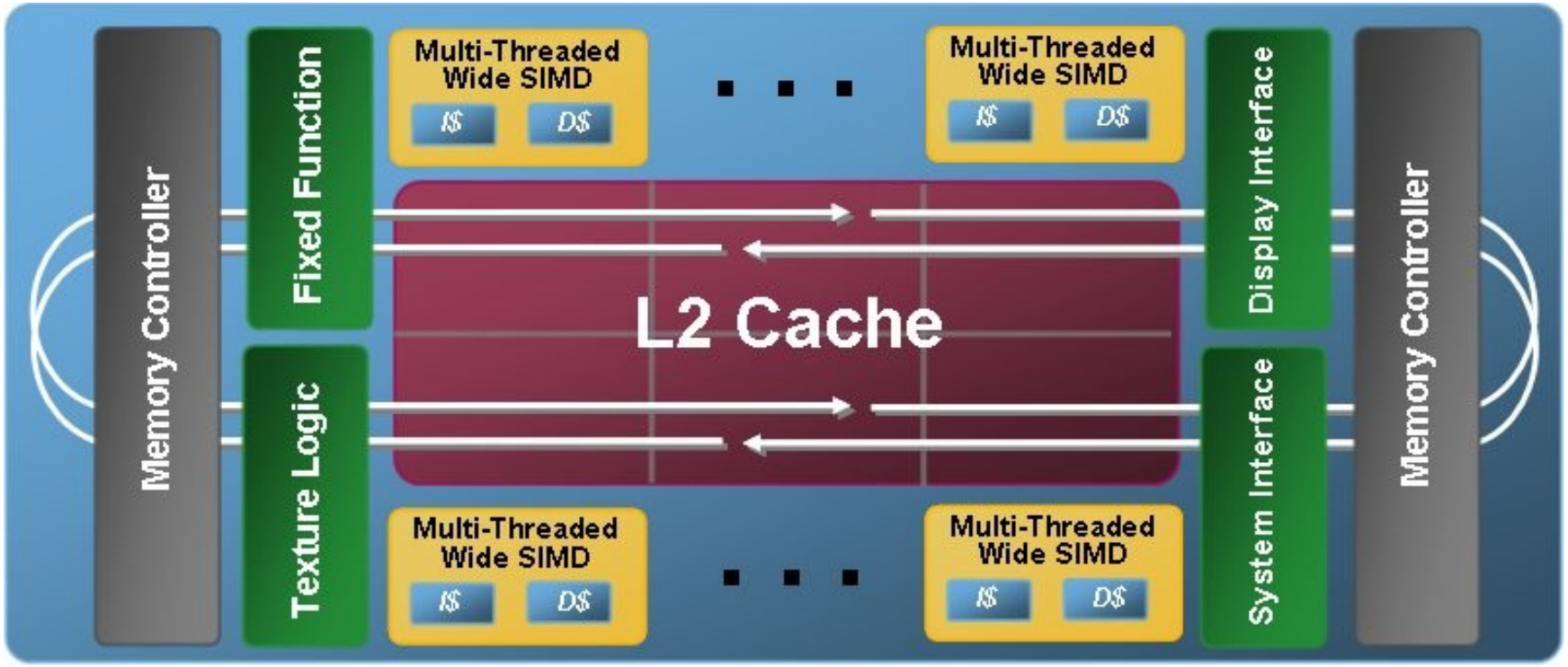
State of the art CPUs

- General-purpose CPUs
  - Power consumption scales quadratic with CPU speed
  - Out-of-order execution – gains in speed and power consumption
  - Multiple hardware threads and CPU cores
    - Enable efficient power management
    - Better suitable if highly parallel algorithms can be found for a certain problem → IBM Cell
    - Cache consistency issues

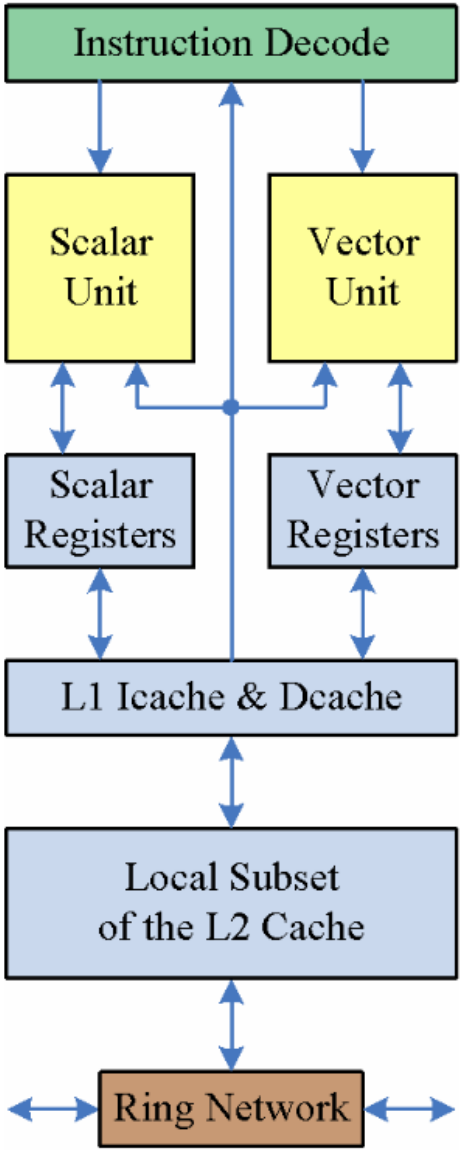
CPUs and GPUs converge



Larrabee architecture



Larrabee Cores



- First release planned with 32 cores
- 4-way multithreaded
- 32kB L1 data & instruction cache
- 256 kB L2 cache subset

- 16 x 32 = 512 bit VPU
  - Load-op instructions
  - Vector arithmetics
  - Scatter/Gather, Unpack operations
  - Vector masks

Larrabee Features

- Explicit cache management
  - Explicit load operation into L1/L2 caches
  - Example: evict streaming data after use, keep other cache lines → avoid thrashing
- No OOO execution
  - Avoid power-intensive hardware
  - Use highly predictable 2-pipelined Pentium cores instead
    - Compiler optimizations instead of runtime speculation

Features (2)

- Interconnect
  - Ring with up to 16 cores
  - 2-way 512 bit bus
  - Cache coherency protocol
  - Multiple memory controllers and fixed function units within the ring to avoid congestion
- Fixed Function Logic
  - Software wherever possible
  - Texture filtering the main point where hardware is needed (up to 40x speed-down otherwise)
- Larrabee software renderer
  - Tile-based to decrease memory demand
  - Dependencies between render targets
  - Cores can switch between front-end and back-end mode
  - 70% of instructions make use of VPU

Evaluation

- Scalability promising
  - 7-10% below optimal scalability for 48 cores
- Memory demand of SW renderer much lower than for immediate (non-tiled) rendering
- Study: can an application balance its load?
  - Too much variance
  - → Need dynamic load balancing.

Larrabee Programming Model

- Larrabee Native C/C++ compiler
  - Static optimization
  - Most existing x86 code compiles w/o modifications
- Pthreads implementation + hardware threading support (Intel Thread Building Blocks)
- Larrabee driver in host
  - Redirect everything L. can't do (e.g., file I/O)
  - Transfer between host and Larrabee nodes
- Claim: SIMD & explicit control good for everything that involves irregular data structures
  - Pointer trees, spatial data structures

Discussion

- *"The 'large' Larrabee in 2010 will have roughly the same performance as a 2006 GPU from Nvidia or ATI." (Peter Glaskowsky, Technology Analyst)*
  - Hint: said at an nVIDIA conference...
- Is Larrabee really different from CUDA/FireStream?
  - Even though Larrabee is x86, we need a dedicated compiler/assembly language for using VPU.
- Is this going to open up a whole new world of bugs and errors?

Discussion (2)

- What is the systems perspective in HPC at all
  - Cryptanalysis (already done)
  - Larrabee as a cheap scalar manycore?
    - Garbage collection
    - Intrusion Detection & other security-related stuff
    - General: offloading work from CPU
  - Can systems software make use of the VPU?